GRADO EN INGENIERÍA TELEMÁTICA

Universidad
Carlos III de Madrid
www.uc3m.es

TRABAJO FIN DE GRADO

Implementación de una herramienta para la asignación eficiente de recursos de ancho de banda en redes ópticas

AUTOR: JOSÉ MORENO ÁLVAREZ

TUTOR: JOSÉ ALBERTO HERNÁNDEZ GUTIÉRREZ

Leganés, Febrero 2014

Agradecimientos:

Estas son mis últimas líneas como estudiante del Grado en Ingeniería Telemática. Dejo atrás años muy duros y difíciles pero, sin duda alguna, los mejores de mi vida. En ellos he conocido gente maravillosa y compartido grandes experiencias que me han permitido llegar a escribir estas líneas. Es por ello que quiero darle las gracias a todas las personas que han compartido conmigo momentos de esta etapa de mi vida.

Principalmente a mis padres, Elías y Teresa, y a mi hermano, Rubén, que para ellos ha sido un gran sacrificio poder ofrecerme esta oportunidad de realizar mis estudios y me han dado todo lo que tengo y todo lo que soy. Siempre han estado y estarán ayudándome y apoyándome y sin ellos no hubiera sido posible. A Charo, que también ha estado siempre animándome y para lo que necesitase.

También a mis amigos del pueblo, que son los mejores y a los cuales he echado de menos estos años. Nos hemos visto menos pero siempre están ahí y hacen que los reencuentros sean aún mejores.

A mis amigos de la Universidad y de la Residencia Fernando Abril Martorell que son lo mejor que me llevo de estos años. Solo por ellos ha merecido la pena el sacrificio y han hecho que el día a día lejos de mi casa haya sido fácil con momentos maravillosos e inolvidables.

Por último, dar las gracias a mi tutor de este Trabajo Fin de Grado, José Alberto, por toda tu ayuda y atención, y por todo lo que he aprendido estos meses, que ha sido mucho.

Gracias a todos de corazón.

Índice general

Índice de figuras

Índice de tablas

Resumen

Este Trabajo Fin de Grado se centra en el estudio de redes de comunicaciones ópticas puesto que desarrollan un papel muy importe en los sistemas de telecomunicaciones actuales. Más en concreto nos centraremos en la Multiplexación por división de longitud de onda (WDM) como método de transmisión en las redes ópticas, puesto que es la tecnología con mejor rendimiento y más utilizada en la actualidad. Estudiaremos los problemas de enrutamiento y asignación de longitudes de onda, RWA, de las redes ópticas actuales a través de sistemas de inecuaciones, que nos permiten resolverlos de una forma óptima.

A la hora de afrontar el estudio de los sistemas ópticos nos parece conveniente dividirlo en tres fases: análisis, dimensionamiento de red y diseño y planificación de red.

En la primera fase nos centramos en el análisis de los distintos tipos de herramientas que podemos utilizar para los problemas basados en inecuaciones, en concreto, MATLAB y Gurobi. Para realizar el análisis implementaremos unos algoritmos que nos permitan obtener las soluciones en estas dos herramientas y compararemos resultados para así decidirnos por la de mejor rendimiento.

Una vez elegida la herramienta que más se adapta a nuestros requisitos, pasamos a la implementación de un algoritmo que resuelve el problema de enrutamiento estático SLE.

La fase de dimensionamiento consiste en la elaboración de un algoritmo basado en Teoría de grafos, que resuelva el problema de asignación de número de longitudes de ondas para cualquier topología de red, dado un tráfico estático.

La última etapa del TFG comprende la creación de algoritmos heurísticos basados en Teoría de colas que nos permitirán diseñar y planificar un sistema de red óptico que recibe unas demandas de tráfico dinámicas.

1.- INTRODUCCIÓN

1.1. Redes ópticas y WDM

1.1.1. Redes ópticas

Se puede denominar a una red óptica como aquella red de telecomunicaciones que realiza alguna de sus funcionalidades a través de medios de comunicaciones ópticos, además del puro transporte de la información.

Una comunicación óptica es toda forma de comunicación que emplea la luz como medio de transmisión.

Un sistema de comunicaciones óptico está formado por un transmisor óptico y un receptor óptico. El transmisor se encarga de codificar el mensaje dentro de una señal óptica o canal, que luego transporta hasta su destino. El receptor, por su parte, tiene la función de reproducir el mensaje recibido a través de la señal óptica.

Los medios modernos más comunes, utilizados en gran variedad de aplicaciones, son los sistemas de comunicación óptica de espacio libre y la fibra óptica.

Los sistemas de comunicación óptica de espacio libre se utilizan generalmente en comunicaciones de "última milla", es decir, pueden funcionar desde distancias cortas hasta distancias de varios kilómetros siempre y cuando haya una línea de visión clara entre el transmisor y el receptor ópticos, y este último pueda realizar la decodificación de la información transmitida.

Por otro lado, la fibra óptica transmite los datos representados en pulsos de luz. Es el canal de comunicación más utilizado en la actualidad para las comunicaciones ópticas, debido a la alta frecuencia que tienen las ondas de luz, que influye en la capacidad de una señal de transportar información, ya que esta aumenta con la frecuencia. El haz de luz es confinado y se propaga por la fibra óptica con un ángulo de reflexión superior al ángulo límite de reflexión total, siguiendo la ley del Snell.

La fibra óptica ofrece muchos beneficios en la transmisión de datos debido a que ofrece un gran ancho de banda, un solo hijo de fibra ofrece un total de 25,000 GHz. Además posee inmunidad a la interferencia electromagnética y tiene unas bajas pérdidas de atenuación (0.5 – 0.2 dB/km).

Los primeros emisores empleados en las fibras ópticas fueron los LEDs, aunque desde el 2007 están prácticamente en desuso. En las redes de comunicaciones ópticas

actuales se utilizan emisores láser. En este tipo de redes se utiliza mayoritariamente la luz infrarroja, puesto que se transmite con menos dispersión y atenuación.

Actualmente, gracias a la ventaja que poseen los repetidores o regeneradores de señal de los sistemas de transmisión de fibra óptica, que pueden estar separados entre sí entorno a unos 100 km, frente a los sistemas electrónicos, que están distanciados aproximadamente 1,5 km; se han podido proporcionar comunicaciones de larga distancia como conexiones transcontinentales y transoceánicas.
Las aplicaciones cada vez más extendidas de los sistemas de fibra óptica se basan en redes de área local (LAN, del inglés Local Area Network) y en redes de área amplia (WAN, del inglés Wide Area Network):

- Las redes de área local están compuestas por un conjunto de computadores que pueden compartir datos, recursos y aplicaciones, estando separados por distancias de hasta pocos kilómetros. Suelen utilizarse en oficinas o campus universitarios permitiendo una transferencia rápida y eficaz de datos entre distintos usuarios y también reduciendo los costes de explotación.

- Las redes de área amplia son similares a las de área local, pero conectan entre sí diversas computadoras separadas por distancias mayores y abarcando varias ubicaciones físicas. Suelen ser construidas por empresas para su uso privado o instaladas por proveedores de internet para ofrecer conexión a sus clientes.

Las redes de fibra óptica surgen como solución a las crecientes demandas de mayor capacidad de transmisión y seguridad por parte de los usuarios y de las empresas operadoras de telecomunicación. A todo esto hay que añadirle un precio lo más económico posible.

Cuando aparece esta demanda de incrementar la capacidad de transmisión entre dos puntos no se disponía de las tecnologías necesarias para satisfacer esta necesidad. Solo quedaba la opción de instalar más fibras entre estos puntos, lo cual suponía una gran inversión de tiempo y dinero, o añadir un más señales multiplexadas por división de tiempo, que también tenía un límite. Fue entonces cuando la multiplexación por división de longitud de onda (WDM) facilitó la solución, pudiendo enviar un gran número de señales por la misma fibra, cada una de ellas por una longitud de onda diferente.

Hasta ahora, en los sistemas basados en arquitecturas eléctricas, cada elemento lleva a cabo su propia restauración de señal y una rotura de una fibra tradicional podría ocasionar el fallo de gran cantidad de sistemas independientes. Sin embargo, los

sistemas de redes ópticas realizan la restauración de la señal en la capa óptica de una forma más rápida y económica que en la capa eléctrica.

En los sistemas que emplean multiplexación eléctrica, cada punto que demultiplexa una señal necesita un elemento eléctrico para cada canal, aunque no estén pasando datos por ese canal. Por el contrario, en una red óptica, solo las longitudes de onda transporten datos necesitan el elemento de red eléctrica, lo que proporciona un gran ahorro de gastos en administración de red y en equipos. También cabe destacar el ahorro económico que supone el mejor aprovechamiento del ancho de banda por parte de los sistemas de redes ópticas con respecto a las fibras simples. A todo esto hay que añadirle el continuo descenso de los precios de los emisores láser.

1.1.2. Multiplexación por división de longitud de onda (WDM)

En el ámbito de las telecomunicaciones, la multiplexación por división de longitud de onda o WDM (del término inglés Wavelength Division Multiplexing), es una tecnología que consiste en combinar, usando generalmente la luz procedente de un láser o un LED, múltiples señales sobre una sola fibra óptica a través de portadoras ópticas de diferente longitud de onda.

El primer sistema WDM apareció alrededor de 1985 y únicamente combinaba dos señales. En la actualidad estos sistemas pueden soportar hasta 160 señales y expandir un sistema de fibra de 10 Gbps hasta una capacidad total de 25,6 Tbps sobre un único par de fibra óptica.

Estos sistemas pueden ser utilizados, dependiendo de su longitud de onda, en tres ventanas de transmisión: primera ventana (de 800 – 900 nm), segunda ventana (de 1260 – 1360 nm) y tercera ventana (de 1500 – 1600 nm).

Los sistemas que implementan la tecnología WDM suelen estar formados por los siguientes componentes:

- **Fuentes de luz:** son fotodiodos emisores de luz o diodos láser.
- **Fibra óptica:** es el medio de transmisión de las señales.
- **Acopladores:** dispositivos que o bien combinan la luz en una fibra óptica, o bien separan la luz en una fibra óptica.
- **Moduladores:** para transmitir información a través de una fibra óptica, los datos han de ser primero codificados, o modulados, en una señal láser.
- **Amplificadores:** son los encargados de regenerar la señal óptica sin convertirla previamente en una señal eléctrica.
- **Conmutadores:** dispositivos que permiten o impiden la transferencia de luz de una guía a otra.

- **Detectores:** se encargan de convertir la señal óptica al dominio eléctrico y de recuperar la información transmitida a través del sistema de comunicaciones ópticas.
- **Filtro óptico:** medio que solo deja pasar a través de él de luz con ciertas propiedades, atenuando o suprimiendo la luz restante.

Y estos sistemas WDM pueden ser de dos tipos: DWDM o CWDM.

La multiplexación por división de longitudes de onda densas o DWDM (del inglés Dense Wavelength Division Multiplexing) es una tecnología de transmisión de señales por medio de fibra óptica utilizando la tercera ventana (1550nm). Este métdo consiste en transmitir varias señales portadoras por una única fibra óptica usando distintas longitudes de onda en cada una de ellas. Cada portadora forma un canal óptico distinto que puede ser tratado independientemente del resto de canales presentes en la fibra óptica y puede contener un tipo de tráfico diferente. Con esto lo que se consigue es poder multiplicar el ancho de banda efectivo de la fibra y facilitar las comunicaciones bidireccionales. Todo esto hace que DWDM sea una técnica de transmisión que resulta muy atractiva para las operadoras de telecomunicaciones, puesto ya que permite el aumento de la capacidad sin necesidad de más cables ni infraestructuras.

Esta tecnología está definida para la ventana de transmisión comprendida entre 1530 nm y 1610 nm, con espaciado entre canales de 0,8 nm y 1,6nm. Se necesitan dos dispositivos complementarios para poder transmitir a través de DWDM, un multiplexor en el lado transmisor y un demultiplexor en el lado receptor.

Por otro lado, la Multiplexación por división en longitudes de onda ligeras o CWDM (del inglés Coarse Wavelength Division Multiplexing) es una tecnología de transmisión de señales por medio de fibra óptica que fue desarrollada principalmente para zonas metropolitanas. Ofrece unos anchos de banda relativamente altos con un coste mucho menor que DWDM, debido a menor complejidad, limitada capacidad y distancia de los componentes ópticos que son utilizados. Se adapta perfectamente a las necesidades de las redes empresariales y metropolitanas de corta distancia.

CWDM se basa en una separación de longitudes de onda, o rejilla, de 2500 GHz (20 nm) dentro del rango comprendido entre 1270 nm y 1610 nm. Puede transportar un número de hasta 28 longitudes de onda dentro de una única fibra óptica monomodo.

Debido a la mayor separación entre longitudes de onda se pueden utilizar láseres con un ancho de banda espectral mayor y no estabilizado, es decir, debido a cambios de temperatura o imperfecciones de fabricación del láser, la longitud de onda central puede desplazarse pero, aún así, estar en banda. Todo esto reduce los costes con respecto a DWDM.

Generalmente, en CWDM se emplean láseres de realimentación distribuida, modulados directamente y que soportan unas velocidades de canal de hasta 2,5 Gbps sobre unas distancias de hasta 80 kilómetros. Los filtros ópticos de banda ancha, multiplexadores y demultiplexadores que utiliza CWDM están basados en TFF (Thin-Film-Filter) o tecnología de película delgada, donde el número de capas del filtro aumenta cuanto menor es el espaciado entre canales, lo que supone una mayor capacidad de integración y una reducción de los costes.

1.2. Introducción a Routing and Wavelength-Assignment (RWA)

En una red óptica WDM, los usuarios finales se comunican entre sí a través de canales ópticos, que se conocen como caminos de luz (lightpaths). Un camino de luz se utiliza para dar soporte a una conexión dentro de una red óptica WDM, y puede abarcar múltiples enlaces de fibra. Un camino de luz debe ocupar la misma longitud de onda en todos los enlaces de fibra óptica que atraviesa. La Ilustración 1 muestra una red donde los caminos de luz se han establecido entre pares de nodos de acceso en diferentes longitudes de onda.

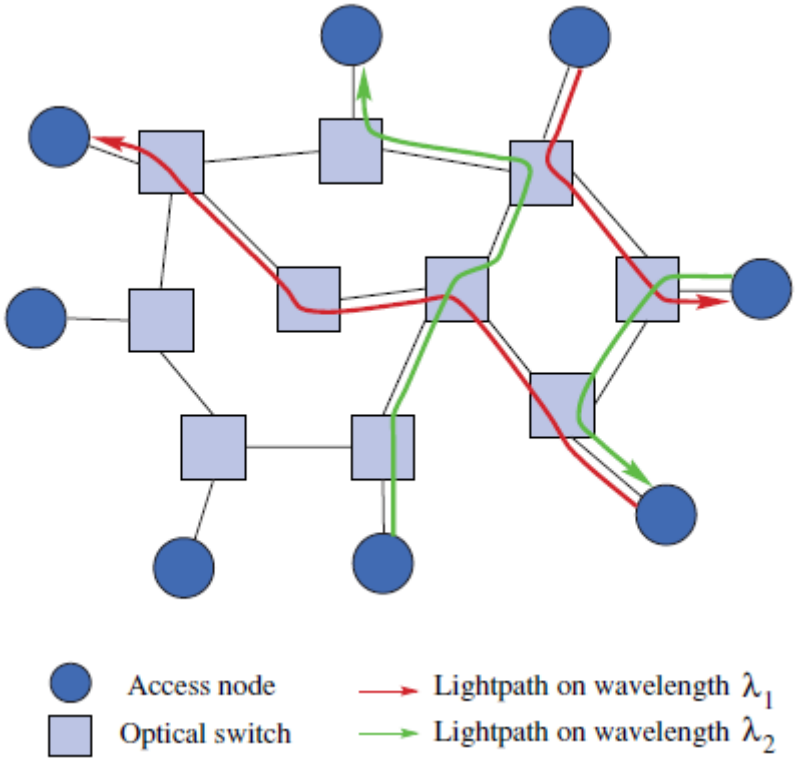


Ilustración 1: Ejemplo de red

Dado un conjunto de conexiones, el problema de la creación de caminos de luz mediante el enrutamiento y la asignación de una longitud de onda para cada conexión se denomina problema "Routing and Wavelength-Assignment", o problema RWA. Generalmente, las solicitudes de conexión pueden ser de tres tipos: estática, dinámica e incremental.

Con el tráfico estático, todo el conjunto de conexiones de luz se conoce con anterioridad, y el problema consiste en establecer cominos de luz para estas conexiones al mismo tiempo que se minimizan los recursos de la red, tales como el número de longitudes de onda o el número de fibras ópticas en el sistema. El problema RWA para el tráfico estático se conoce como problema SLE (del inglés Static Lightpath Establishment).

Para el caso del tráfico dinámico, se establece un camino de luz para cada solicitud de conexión a medida que llega, y este camino se libera tras pasar una cierta cantidad finita de tiempo.

Con el tráfico incremental, las solicitudes de conexión llegan secuencialmente, se establece un camino de luz para cada conexión y este camino permanece en la red indefinidamente.

El objetico en los casos de tráfico incremental y tráfico dinámico, es la creación de caminos de luz y asignar longitudes de onda de una manera que se minimice la cantidad de bloqueos de conexión, o que maximice el número de conexiones que se establecen en la red en cualquier momento. Este problema se define como problema DLE (del inglés Dynamic Lightpath Establishment).

El problema SLE se puede formular como un programa lineal entero mixto. Para hacer este problema más tratable, SLE se puede dividir en dos subproblemas, uno de enrutamiento y otro de asignación de longitud de onda, que se pueden resolver por separado. Se emplean algoritmos de aproximación para resolver el problema SLE para redes grandes, y algoritmos "graph-coloring" para asignar longitudes de onda a los caminos de luz, una vez que estos caminos han sido enrutados correctamente. Por otro lado, el problema DLE es más fácil de resolver, y por lo tanto, se emplean métodos heurísticos tanto para resolver el subproblema de enrutamiento como para el subproblema de asignación de longitud de onda.

Para el subproblema de enrutamiento hay tres enfoques básicos: enrutamiento fijo, enrutamiento fijo alternativo y enrutamiento adaptativo. Entre estos tres enfoques, el enrutamiento fijo es el más simple, mientras que el enrutamiento adaptativo alcanza un mejor rendimiento. El enrutamiento fijo alternativo ofrece una solución intermedia entre la complejidad y el rendimiento.

En el caso del subproblema de asignación de longitud de onda, se han propuesto una serie de heurísticos: asignación aleatoria de longitud de onda, mejor ajuste, menos utilizado, más utilizado, con mayor carga, con menor carga, pérdida de capacidad relativa, reserva de longitud de onda y protección de umbral, entre otros. Actualmente, el algoritmo que ofrece mejor rendimiento es el de pérdida de capacidad relativa o RCL (del inglés Relative Capacity Loss). Sin embargo, el algoritmo RCL es relativamente costoso de implementar en una red óptica y puede presentar alguna sobrecarga de control significativa.

1.3. Notación

A lo largo de este proyecto vamos a utilizar una notación denominada "link-demand-path-identifier-based". Esta notación es compacta y nos permite mostrar únicamente los objetos necesarios. Si bien es cierto que la nueva notación puede parecer poco intuitiva al principio, una vez que lo hayamos explicado veremos que es muy útil para capturar, formular y comprender los problemas de flujo de red, así como para hacer manipulaciones algebraicas sobre los problemas formulados.

A todos los pares de demandas que tienen un volumen de demanda distinto de cero, se le asignan índices entre 1 y el número total de pares de demanda. Ahora consideremos el siguiente ejemplo de la Ilustración 2, donde podemos etiquetar y asignar los pares de demandas de la siguiente manera:

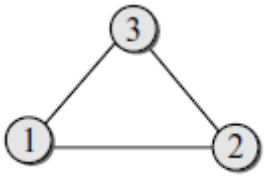


Ilustración 2: Topología triángulo

par demanda $\langle 1,2 \rangle$ ←→ etiqueta de demanda 1

par demanda $\langle 1,3 \rangle$ ←→ etiqueta de demanda 2

par demanda $\langle 2,3 \rangle$ ←→ etiqueta de demanda 3

Esto significa que al par demanda $\langle 1,2 \rangle$ (demanda que va desde el nodo 1 hasta el nodo 2) le asignaremos la etiqueta de demanda 1. Así será la segunda demanda aquella que tiene como origen el nodo 1 y como destino el nodo 3. La tercera demanda tiene como fuente el nodo 2 y como destino el nodo 3.

En general, utilizaremos la notación "*D*" para indicar el número total de pares de demandas en una red que tienen volúmenes de demanda positivos, y el índice "*d*" para

etiquetar estas demandas. Por lo tanto, en el ejemplo anterior D = 3 y d =1, 2, 3. Del mismo modo, los enlaces que existen en la red se etiquetan desde 1 hasta el número total de enlaces. Como en el caso de un par demanda, si no existe enlace entre dos nodos específicos, no necesitamos añadirlo a la formulación del problema. Para el mismo ejemplo, tendríamos:

enlace 1-2 ← → etiqueta de enlace 1

enlace 1-3 ← → etiqueta de enlace 2

enlace 2-3 ← → etiqueta de enlace 3

Queda así reflejado el enlace que une el nodo 1 con el nodo 2, como enlace 1. El segundo enlace será aquel que une el nodo 1 con el nodo 3. Y el tercer enlace será el que une el nodo 2 con el nodo 3.

Para el caso de los enlaces, emplearemos la notación "E" para referirnos al número total de enlaces reales en la red, y el índice "e" para etiquetar los enlaces. En este caso tenemos E = 3 y e = 1, 2, 3.

En cuanto al tema de los volúmenes de demanda, utilizaremos la notación "h" para identificar que se trata de un volumen de demanda, y el subíndice "d" para referenciar a que demanda corresponde. Siguiendo con el ejemplo anterior, "h_1" identificaría al volumen de demanda que existiría para la demanda 1 (cantidad de información a transferir desde el nodo 1 hasta el nodo 2). Así "h_2" y "h_3" harían referencia a los volúmenes de las demandas 2 y 3 respectivamente.

Respecto a las capacidades de los enlaces, seguiremos la línea de los volúmenes de demanda, empleando la notación "c" para referirnos a una capacidad, y el subíndice "e" para identificar el enlace correspondiente. Para el ejemplo que venimos utilizando, "c_1", "c_2" y "c_3" representan la capacidad de los enlaces 1, 2 y 3 respectivamente.

Pasaremos ahora a analizar como denotar los identificadores de los caminos. Puesto que tenemos un identificador de par demanda, lo utilizaremos como primer subíndice en una variable de ruta de flujo, y el segundo subíndice lo emplearemos para etiquetar el camino de un par demanda particular. Esto es similar a lo hablado anteriormente para pares de demanda y enlaces, numerando los caminos candidatos para un par demanda desde 1 hasta el número total de caminos candidatos para ese par demanda. El número total de caminos candidatos para la demanda "d" se denota por "P_d" y los caminos serán etiquetados con el índice "p". Por ejemplo, para la demanda $\langle 1,2 \rangle$ identificada con la etiqueta d = 1, tenemos P_1 = 2, siendo los caminos candidatos 1-2 y 1-3-2, etiquetados con p = 1, 2 respectivamente (como segundo subíndice en la variable de ruta de flujo). Por tanto, se identifican como P_{11} y P_{12}, es decir, como rutas

de acceso número 1 y número 2 de la demanda 1. Podemos, a partir de los caminos de la Ilustración 3, denotar las variables de flujo de la siguiente manera:

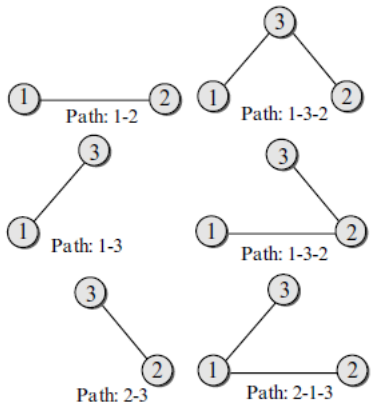


Ilustración 3: Distintos caminos de la topología triángulo

$x_{11} \rightarrow$ flujo para el primer camino candidato de la demanda 1

$x_{12} \rightarrow$ flujo para el segundo camino candidato de la demanda 1

$x_{21} \rightarrow$ flujo para el primer camino candidato de la demanda 2

$x_{22} \rightarrow$ flujo para el segundo camino candidato de la demanda 2

$x_{31} \rightarrow$ flujo para el primer camino candidato de la demanda 3

$x_{32} \rightarrow$ flujo para el segundo camino candidato de la demanda 3

Y a partir de todo lo mencionado hasta ahora, podemos definir la función objetivo "*F*" que, como veremos a continuación, va a consistir en minimizar el coste total del enrutamiento.

1.4. Presentación de topologías de red

En este apartado haremos una breve presentación de las topologías de red que utilizaremos a lo largo de este proyecto para poner en práctica los experimentos realizados.

Topología triángulo

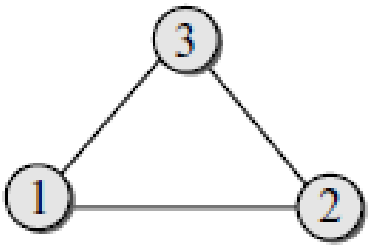


Ilustración 2: Topología triángulo

Se trata de una topología de red bastante sencilla que utilizaremos para explicar algún ejemplo de forma sencilla y para hacer comparaciones con los distintos recursos utilizados.

Topología pentágono

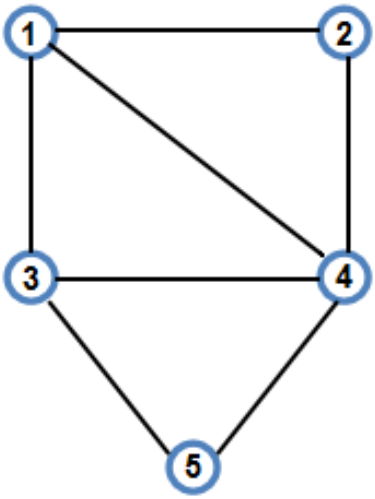


Ilustración 4: Topología pentágono

Esta topología de red es algo más compleja que la red anterior y la emplearemos como punto intermedio para las comparaciones entre la topología triángulo y el resto de sistemas de mayor extensión.

Topología de prueba

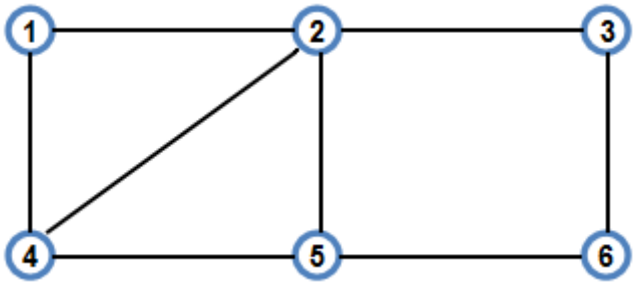


Ilustración 5: Topología de prueba

Se trata de una topología de red inventada con poca extensión y que utilizaremos para explicar algún caso teórico de forma más entendible así como para generar resultados y comparaciones más profundas.

Topología RedIRIS

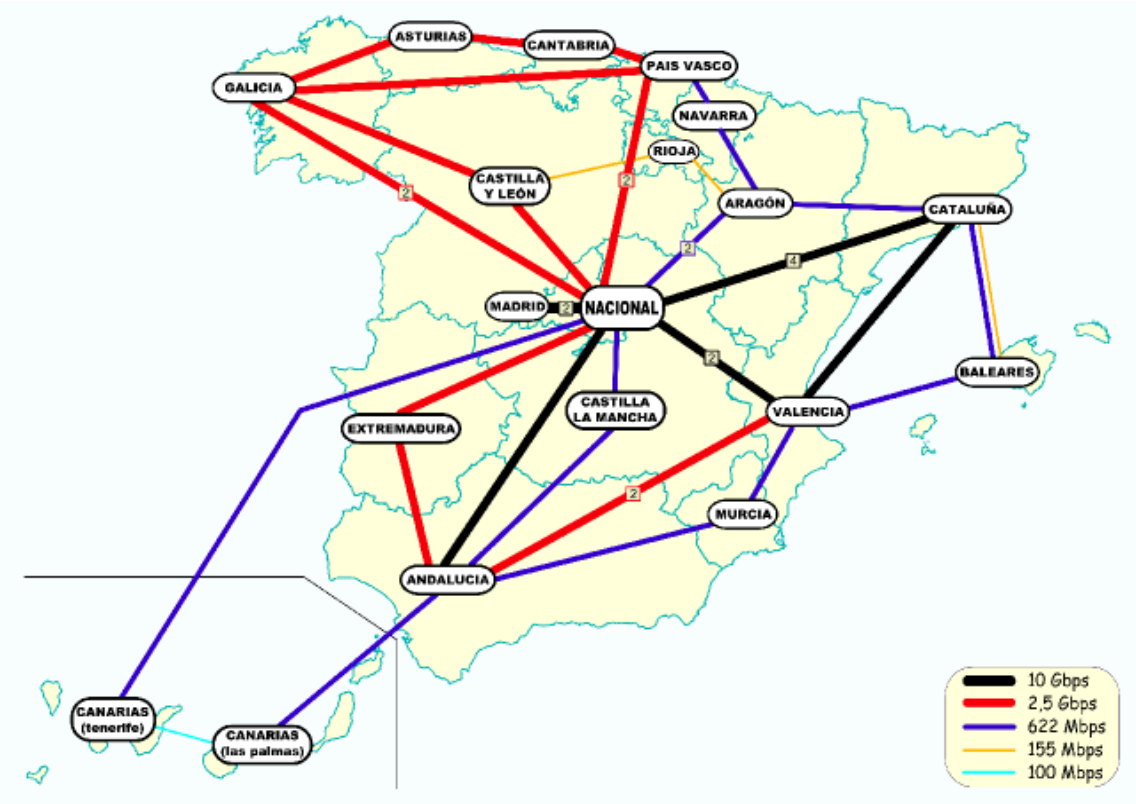


Ilustración 6: Topología RedIRIS

Topología de red basada en la RedIRIS, que es la red académica y de investigación del territorio nacional español. Esta red provee de servicios avanzados de comunicaciones a la comunidad académica y científica nacional. Está financiada por el Ministerio de

Economía y Competitividad y está compuesta por más de 450 instituciones afiliadas, principalmente instituciones universitarias y de investigación.

Topología NSFNet

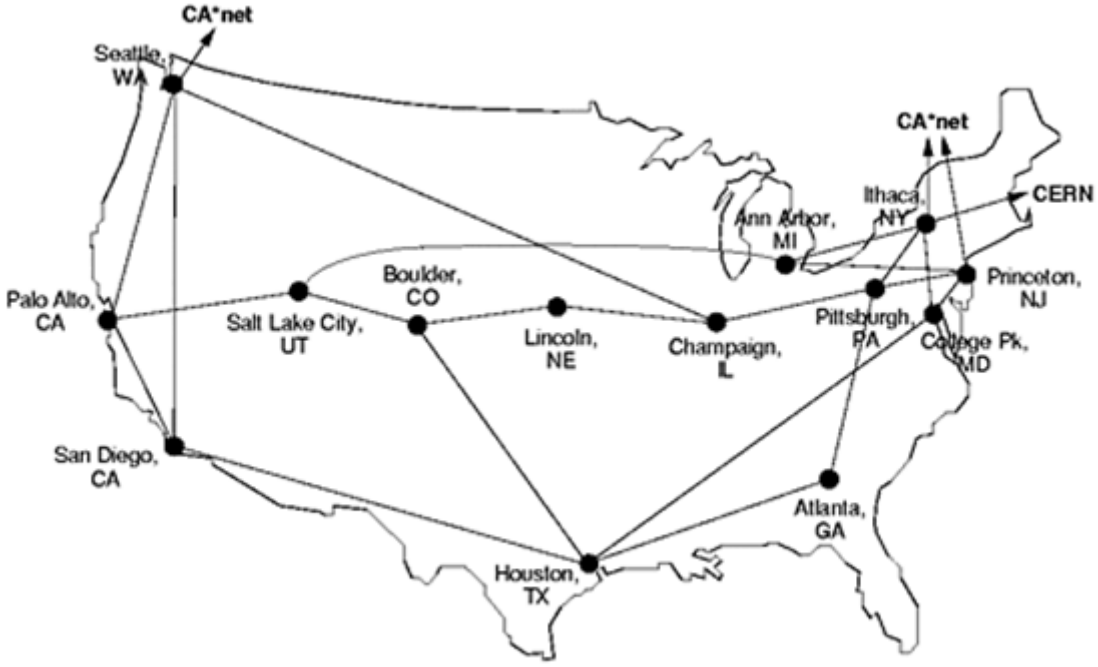


Ilustración 7: Topología NSFNet

Topología basada en la red NSFNet, que es una red de área amplia desarrollada por la National Science Foundation (NSF). NSFNet remplazó a la red ARPANet como la red principal del gobierno estadounidense que conecta las universidades y los centros de investigación. Posteriormente la NFS desmanteló esta red y la reemplazó con un backbone de Internet comercial.

Topología de Telefónica

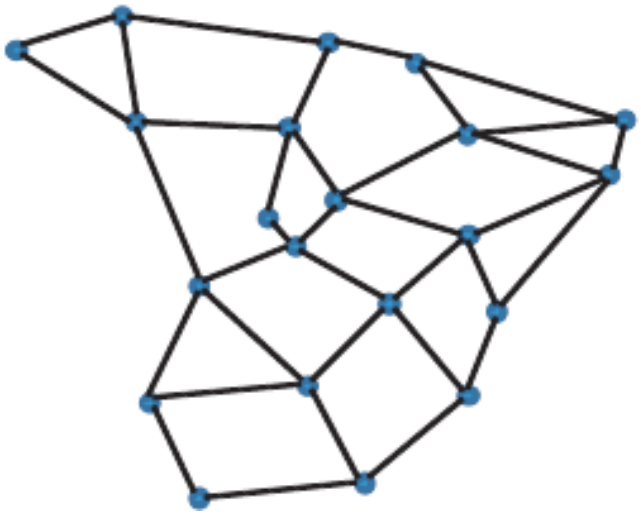


Ilustración 8: Topología Telefónica

Se trata de la topología de red óptica de Telefónica en España. Es una red de gran extensión que emplearemos para comentar resultados de los experimentos a través de gráficos y comprarlos con otras topologías.

1.5. Planteamiento del problema inicial

Tras haber explicado cómo funcionará la notación, podemos pasar a definir el problema inicial. Este ejercicio va a consistir en formular una serie de restricciones tanto de demanda como de capacidad para poder definir una función objetivo que minimice el coste total del enrutamiento.

Podemos, por lo tanto, pasar a formular las restricciones de demanda a partir del ejemplo de la Ilustración 2, y con los posibles caminos que tomábamos de la Ilustración 3. Quedarían de la siguiente manera:

$$x_{11} + x_{12} = h_1$$

$$x_{21} + x_{22} = h_2$$

$$x_{31} + x_{32} = h_3$$

Como observamos, el volumen de demanda 1 "h_1" está formado por el flujo de tráfico que atraviesa el primer camino candidato para la demanda 1, más el flujo que atraviesa el segundo camino candidato para dicha demanda. De la misma forma, "h_2" y "h_3" están compuestos por la suma de los volúmenes de demanda que pasan por sus propios caminos candidatos. También podemos fijarnos en que van a existir tantas restricciones de demanda como número de demandas de tráfico haya en la red, que en este ejemplo son tres.

Por otro lado, existirán tantas restricciones de capacidad como número de enlaces contenga la red. Quedando de la siguiente forma:

$$x_{11} + x_{22} + x_{32} \leq c_1$$

$$x_{12} + x_{21} + x_{32} \leq c_2$$

$$x_{12} + x_{22} + x_{31} \leq c_3$$

Podemos observar que las capacidades de los enlaces tienen que ser mayor o igual que la suma de todos los flujos de tráfico que atraviesan ese enlace. Así, la capacidad del

enlace 1 será mayor o igual que la suma compuesta por el flujo de tráfico de la demanda 1 para su primer camino candidato, más el flujo de la demanda 2 para su segundo camino candidato y más el flujo de la demanda 3 para su segundo camino candidato. Con el resto de costes sucede exactamente lo mismo.

Suponiendo que el objetivo de nuestro problema inicial es minimizar el coste total del enrutamiento, si asumimos que el coste de encaminar cada flujo de tráfico a lo largo de su trayectoria es 1, tendríamos la función objetivo o coste total de encaminamiento siguiente:

$$F = x_{11} + 2x_{12} + x_{21} + 2x_{22} + x_{31} + 2x_{32}$$

1.6. Estructura del TFG

En este apartado comentaremos la estructura que seguirá esta memoria para afrontar todos los puntos del proyecto.

Después de este capítulo de introducción pasaremos a hablar, en el capítulo 2, sobre la parte de análisis del proyecto. Aquí explicaremos las dudas iniciales sobre la resolución de nuestro primer problema, explicaremos los algoritmos creados para la solución de este problema y compararemos los resultados obtenidos. Los algoritmos implementados tienen la siguiente estructura:

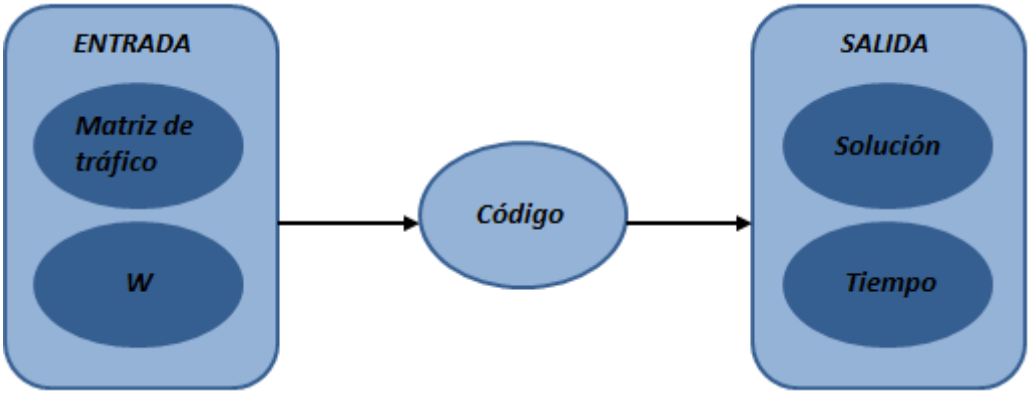


Ilustración 9: Estructura algoritmos 1 y 2

En la sección 3, trataremos el problema SLE de enrutamiento estático. Explicaremos el problema con su notación y sus ecuaciones. Posteriormente expondremos los cambios necesarios en las ecuaciones para poder implementar un algoritmo que pueda resolver el problema utilizando Gurobi y explicaremos los resultados obtenidos. El algoritmo sigue la estructura:

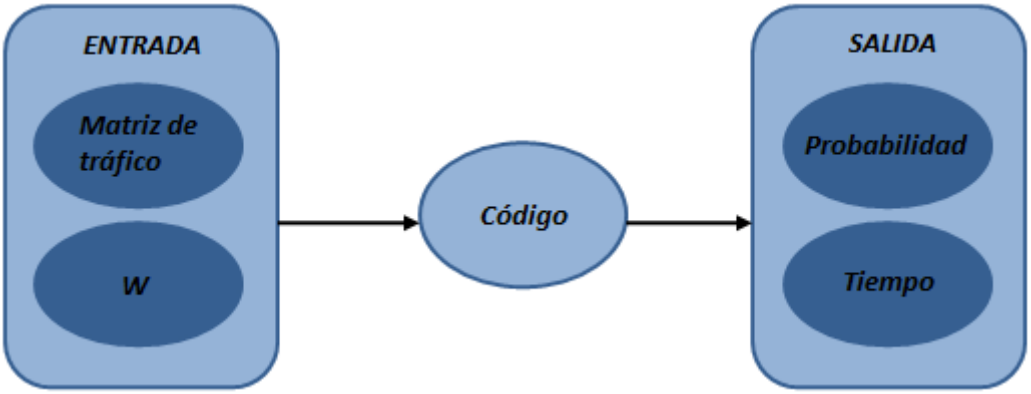


Ilustración 10: Estructura algoritmo 3

En el cuarto capítulo, afrontaremos el problema de dimensionamiento de red. Este problema consiste en la asignación de longitudes de ondas y está basado en la Teoría de grafos. Explicaremos la Teoría de grafos y como la adaptamos a nuestras necesidades para poder implementar un algoritmo que resuelva nuestro problema. Este algoritmo tiene la siguiente estructura:

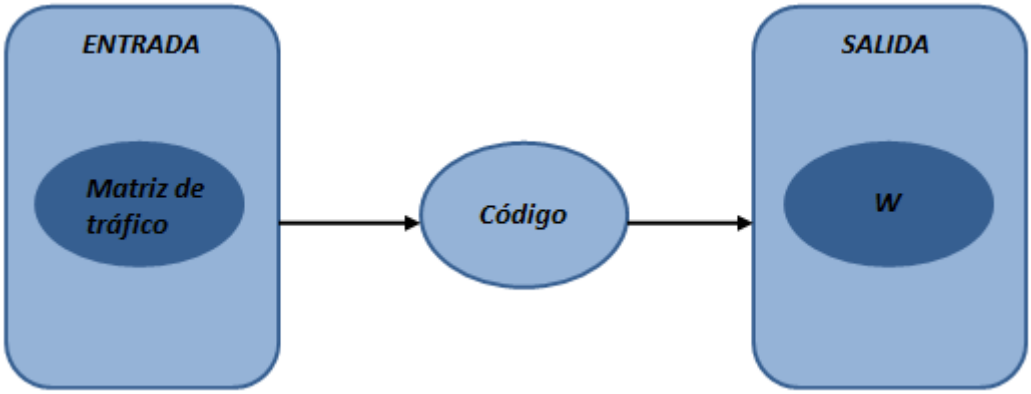


Ilustración 11: Estructura algoritmo 4

En el quinto apartado, explicaremos la implementación de dos algoritmos heurísticos basados en la Teoría de colas. Para ello haremos una breve introducción a la heurística y a la Teoría de colas y comentaremos los resultados de estos dos algoritmos. Los dos programas llevarán la siguiente estructura:

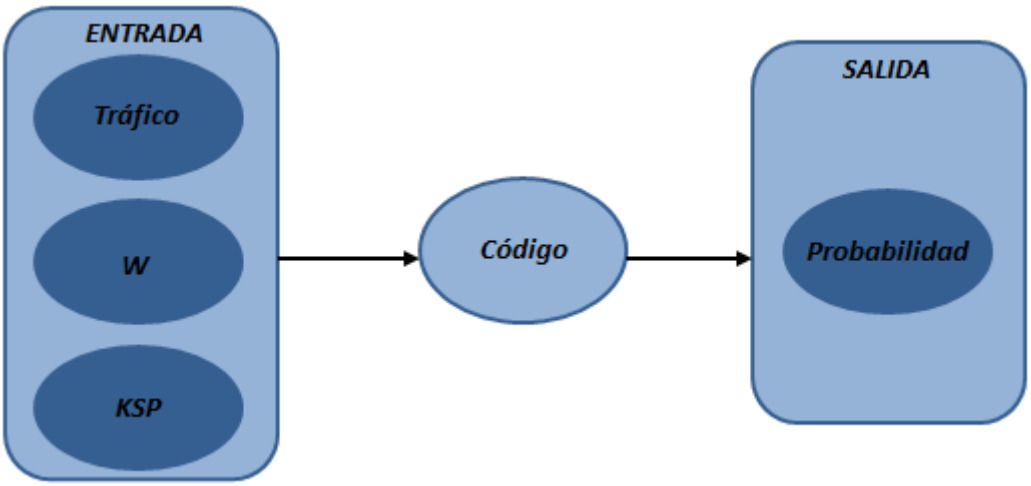


Ilustración 12: Estructura algoritmos 5 y 6

En el capítulo 6, haremos una conclusión global de todos los algoritmos que hemos implementado para resolver cada uno de nuestros problemas durante este TFG. Pondremos en común todas las conclusiones llevadas a cabo en cada uno de ellos.

Al final del documento vamos a incluir un apartado de anexos donde incluiremos el presupuesto del proyecto, un resumen en inglés y un apartado de ejemplo de código.

En el apartado de presupuesto explicaremos los costes deducidos de los componentes hardware y software y también los costes de personal. Luego pondremos en común estos costes y veremos el presupuesto total de este TFG.

El apartado de resumen en inglés contiene una recopilación de lo más destacado del proyecto redactado en lengua inglesa.

En la sección de ejemplo de código adjuntamos una pequeña parte del código de alguno de nuestros algoritmos para que puedan servir de referencia si es necesario.

2.- ANÁLISIS

A partir del planteamiento del problema inicial como un sistema de inecuaciones, se nos plantean varias vías para afrontar y resolver este problema. En este apartado explicaremos cuáles son los distintos tipos de herramientas que contemplamos, implementaremos algoritmos que resuelvan el problema con estas herramientas y explicaremos los criterios de la elección final de uno de ellas.

Principalmente nos basaremos en dos propuestas para dar con la solución a nuestro problema. Estas herramientas son: MATLAB y GUROBI.

2.1. MATLAB

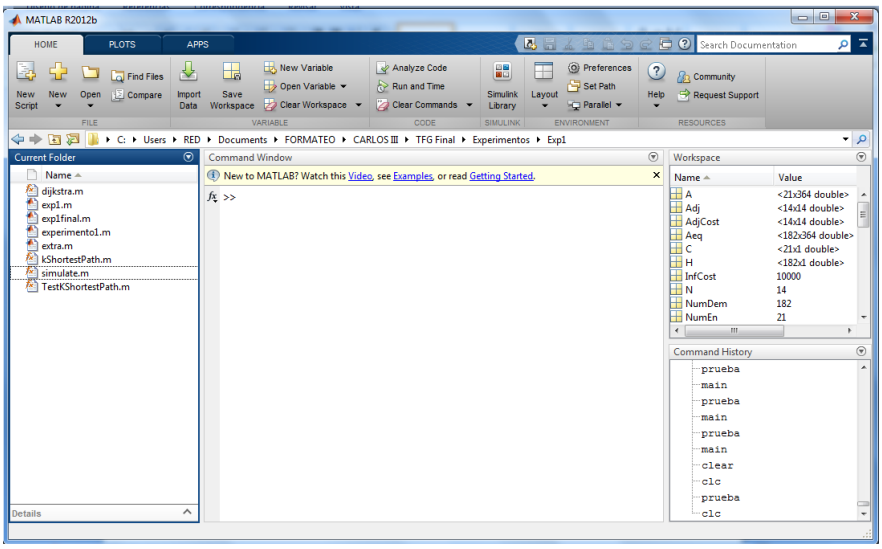
Ilustración 13: Logotipo MATLAB

2.1.1. Introducción a MATLAB

MATLAB, que toma nombre de abreviar el término "MATrix LABoratory", es un paquete de software matemático que ofrece un entorno de desarrollo distribuido o IDE (del inglés Integrated Development Environment), es decir, un programa informático formado por un conjunto de herramientas de programación. Posee un lenguaje de programación propio, llamado lenguaje M, y proporciona cálculo numérico, computación de matrices y representación de datos y gráficos en un entorno de trabajo muy cómodo para sus usuarios. Se encuentra disponible para los sistemas operativos y plataformas Windows, Mac OS X, Unix y GNU/Linux.

Cleve Moler originó la primera versión de MATLAB en 1984, a partir de los paquetes de cálculo matricial LINPACK y EISPACK. En las versiones posteriores se han ido añadiendo diferentes librerías, llamadas Toolboxes, especializadas en áreas científicas concretas y que han hecho que MATLAB se haya convertido en una herramienta instructora básica

en los entornos universitarios. También es muy utilizada en los entornos de investigación e industriales para resolver problemas prácticos y cálculos de ingeniería.



Ilustracion 14: Ventana de comandos de MATLAB

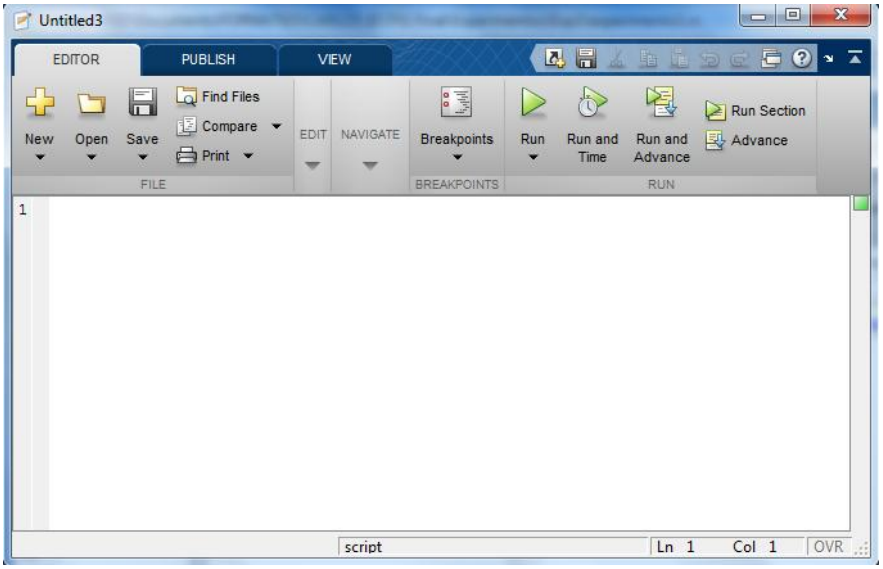


Ilustración 15: Editor de MATLAB

MATLAB funciona como un programa intérprete de comandos, es decir, procesa de forma secuencial una lista de comandos previamente definidos por MATLAB o por el usuario, y obtiene de manera inmediata los resultados. La serie de comandos puede ser escrita en la ventana de comandos, si se trata de un número pequeño de instrucciones, o bien en un archivo con extensión .m formando así un programa. Si los datos se escriben en la ventana de comandos, estos serán procesados tras la orden de ejecución ENTER. Si por el contrario, los comandos son escritos en un fichero, podemos procesarlos desde el editor propio de MATLAB, ejecutando la opción "Run" o bien desde la ventana de comandos escribiendo el nombre del archivo.

Este programa tiene una sintaxis propia, por lo tanto, para poder ejecutar las expresiones es necesario respetar estas reglas sintácticas, que suelen ser bastante parecidas a las de otros lenguajes de programación.

2.1.2. Función fmincon

La función fmincon nos permite encontrar el valor escalar mínimo que devuelve una función no lineal de varias variables y que está sujeta a una serie de restricciones. Esta función pertenece al Toolbox de Optimización, que está formado por un conjunto de funciones que amplían la capacidad de computación numérica de MATLAB, en concreto, logran maximizar o minimizar funciones no lineales generales.

Por lo tanto, fmincon resuelve problemas de optimización no lineal con restricciones. La función no lineal multivariable toma el nombre de función objetivo y debe ser implementada por el usuario para que se lleven a cabo los cálculos que obtengan el valor escalar que devuelve la función.

El propósito de la función fmincon es calcular el mínimo de *f(x)* sujeta a las siguientes restricciones:

$$c(x) = 0$$

$$c_{eq}(x) = 0$$

$$A \cdot x = b$$

$$A_{eq} \cdot x \leq b_{eq}$$

$$lb \leq x \leq ub$$

donde A y A_{eq} son matrices; x, b, b_{eq}, lb y ub son vectores; c(x) y c_{eq}(x) son funciones que devuelven vectores y f(x) que es la función objetivo.

La función fmincon tiene una gran variedad tanto de parámetros de entrada como de salida, por lo que nos centraremos en explicar los parámetros que utilizaremos para resolver el problema inicial. El resto de parámetros pueden ser consultados mediante el comando "help fmincon" desde la ventana de comandos de MATLAB.

Tendremos entonces que invocar a la función fmincon en MATLAB, con los siguientes parámetros de entrada y salida:

$$[x, fval, exitflag] = fmincon(fun, x0, A, b, Aeq, beq, lb, lu)$$

Empezaría con los valores especificados en el vector de inicio "x0" y encontraría un valor mínimo del vector "x" para la función objetivo descrita en "fun", tal que cumpla las restricciones $A \cdot x = b$ y $A_{eq} \cdot x \leq b_{eq}$, y que dicho valor se encuentre comprendido entre "lb" y "ub". La función además devuelve en "fval" el valor de la función objetivo "fun"

evaluada en el vector "x", y la condición de salida "exitflag". Esta condición de salida "exitflag" pude tomar valores: mayores que 0 (si la función f(x) converge en la solución "x"), igual a 0 (si ha alcanzado el número máximo de iteraciones de la función fmincon) o menor que 0 (si la función f(x) no converge en una solución).

A través del siguiente ejemplo se entenderá mejor la forma de empleo de esta función:

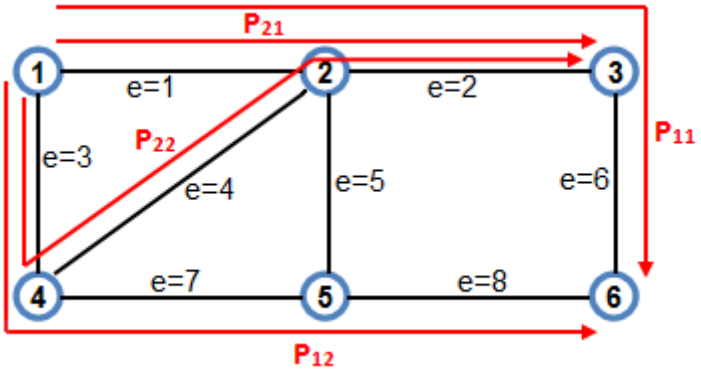


Ilustración 16: Topología de prueba con demandas

Como podemos ver, en este ejemplo tenemos dos demandas:

- d = 1, demanda con origen nodo 1 y destino nodo 6
- d = 2, demanda con origen nodo 1 y destino nodo 3

Ambas demandas tienen dos caminos posibles ($P_d = 2$, con d = 1, 2):

- Para d = 1, tenemos $P_1 = 2$ cuyos caminos son P_{11} y P_{12}
- Para d = 2, tenemos $P_2 = 2$ cuyos caminos son P_{21} y P_{22}

Suponiendo ahora que la demanda de tráfico para demanda 1 sea $h_1 = 5$ unidades de tráfico, y para la demanda 2, $h_2 = 4$ unidades; y que la capacidad de todos los enlaces es igual a 3 unidades de tráfico ($c_e = 3$, con e = 1, 2, 3, 4, 5, 6, 7, 8), tendremos las siguientes restricciones de demanda y capacidad:

$$x_{11} + x_{12} = 5$$
$$x_{21} + x_{22} = 4$$

$$x_{11} + x_{21} \leq 3$$
$$x_{11} + x_{21} + x_{22} \leq 3$$
$$x_{12} + x_{22} \leq 3$$
$$x_{22} \leq 3$$
$$0 \leq 3$$
$$x_{11}$$
$$x_{12} \leq 3$$
$$x_{12} \leq 3$$

Donde podemos observar que la quinta ecuación de capacidad es $0 \leq 3$, debido a que por el enlace e $= 5$ no pasa ningún flujo de demanda.

Por tanto, tendremos las matrices:

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 5 \\ 4 \end{bmatrix}$$

$$A_{eq} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad B_{eq} = \begin{bmatrix} 3 \\ 3 \\ 3 \\ 3 \\ 3 \\ 3 \\ 3 \\ 3 \end{bmatrix}$$

Asumiendo que el coste de encaminar cada flujo de tráfico a lo largo de su trayectoria es 1, tendríamos la función objetivo:

$$F = 3x_{11} + 3x_{12} + 2x_{21} + 3x_{22}$$

Que podemos traducir, para ser usada por fmincon, de la siguiente forma:

$$fun = sum(sum(A_{eq}).* x)$$

donde "x" es el vector devuelto por fmincon.

Si queremos que nuestro ejercicio parta de los valores iniciales de los "x_{pd}" iguales a cero, crearíamos el vector:

$$x0 = [0\ 0\ 0\ 0]$$

Supongamos ahora que existe un límite inferior de los valores de x_{pd}, lb, igual a cero y no hay límite superior, ub, quedarían los vectores:

$$lb = [0\ 0\ 0\ 0]$$

$$ub = []$$

Con todo estos argumentos la llamada a la función quedaría de la siguiente manera:

$$[x, fval, exitflag] = fmincon(fun, x0, A, B, A_{eq}, B_{eq}, lb, ub)$$

2.2. GUROBI OPTIMIZER

Ilustración 17: Logotipo de Gurobi

2.2.1. Introducción a Gurobi Optimizer

Gurobi es un software comercial para la optimización de problemas de programación matemática. Incluye los siguientes solucionadores: solucionador de programación lineal (LP del inglés Linear Programming), solucionador de programación cuadrática (QP, Quadratically Programming), solucionador de programación cuadráticamente restringida (QCP, Quadrat Constrained Programming), solucionador de programación linear entera mixta (MILP, Mixed-Integer Linear Programming), solucionador de programación cuadrática entera mixta (MIQP, Mixed-Integer Quadratic Programming) y solucionador de programación cuadráticamente restringida entera mixta (MIQCP, Mixed-Integer Quadratically Constrained Programming).

Los solucionadores de Gurobi han sido diseñados desde cero para aprovechar las arquitecturas modernas y los procesadores multi-núcleo, usando las implementaciones más avanzadas de los últimos algoritmos.

Este optimizador va más allá del rendimiento de una solución rápida y fiable para proporcionar una amplia gama de interfaces y acceso a lenguajes de modelado estándar.

Gurobi es accesible desde entornos de desarrollo distribuidos gracias a la gran variedad de bibliotecas que tiene disponible. Tiene disponible actualmente interfaces para los siguientes lenguajes de programación: C, C++, Java, Microsoft .NET, Python, MATLAB y R.

Nosotros nos centraremos en la interfaz de usuario para MATLAB, que comentaremos a continuación.

2.2.2. Interfaz de Gurobi Optimizer para MATLAB

La interfaz Gurobi MATLAB permite contruir un modelo de optimización, pasar este modelo a Gurobi, y obtener el resultado de la optimización, todo ello desde el entorno de desarrollo de MATLAB.

La interfaz Gurobi MATLAB es bastante concisa. Se compone únicamente de cuatro funciones MATLAB: gurobi(), gurobi_read(), gurobi_write() y gurobi_setup().

Para el desarrollo del proyecto solamente utilizaremos la función gurobi(). Por lo tanto, nos centraremos en explicar en qué consiste.

La función gurobi() tiene dos argumentos de entrada y uno de salida, todos ellos son variables struct de MATLAB. Cada argumento está formado por varios campos. Para invocar desde MATLAB a esta función, lo haríamos de la siguiente manera:

result = gurobi(model,params)

El primer argumento "model" contiene el modelo de optimización que hay que resolver. Contiene varios campos que representan las diversas partes del modelo de optimización. Varios de estos campos son opcionales. Los más importantes para la correcta resolución del problema inicial son:

- A : matriz de restricción lineal.
- obj : vector objetivo lineal. Se debe especificar un valor para cada columna de A.
- sense : sentidos de las restricciones lineales. Los valores permitidos son '=', '<' o '>'. Se debe especificar un valor para cada fila de A, o un único valor para especificar que todas las restricciones tienen el mismo sentido.
- rhs : vector del lado derecho de las restricciones lineales. Se debe especificar un valor para cada fila de A.
- lb : vector de límite inferior. Se debe especificar un valor para cada columna de A. Campo opcional.
- ub : vector de límite inferior. Se debe especificar un valor para cada columna de A. Campo opcional.
- modelsense : sentido de la optimización. Los valores permitidos son 'min' (minimizar) o 'max' (maximizar). Campo opcional, por defecto toma el valor 'mim'.

El resto de campos son opcionales y pueden ser consultados en la documentación de Gurobi.

El segundo argumento "params" es opcional y contiene un conjunto de parámetros Gurobi que permite al usuario modificar el comportamiento predeterminado de los algoritmos de optimización. Se utilizan para casos muy concretos. Estos campos pueden ser consultados en la documentación de Gurobi.

El argumento de salida "result" contiene los diversos resultados de la optimización almacenados en sus campos. Estos resultados dependen del tipo de modelo a resolver y del estado de la optimización. Los campos que más nos interesa comentar son:

- status : estado de la optimización devuelto en una cadena de caracteres. El resultado deseado es 'OPTIMAL', que indica que se ha encontrado una solución

óptima para el modelo. Puede haber otros estados, por ejemplo, si el modelo no tiene una solución factible devolvería 'INFEASIBLE'. El resto de valores pueden ser consultados en la documentación.

- objval : valor objetivo de la solución calculada.
- runtime: tiempo de ejecución (en segundos) transcurrido para la optimización.
- x : solución calculada. Esta matriz contiene una entrada para cada columna de A.

El resto de campos pueden ser consultados en la documentación de Gurobi.

Utilizaremos el ejemplo de la Ilustración 16 para ilustrar fácilmente cómo podemos utilizar esta función. Quedando los elementos necesarios de la siguiente forma:

$$A = \begin{bmatrix} 1\,0\,1\,0 \\ 1\,0\,1\,1 \\ 0\,1\,0\,1 \\ 0\,0\,0\,1 \\ 0\,0\,0\,0 \\ 1\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,1\,0\,0 \\ 1\,1\,0\,0 \\ 0\,0\,1\,1 \end{bmatrix} \quad rhs = \begin{bmatrix} 3 \\ 3 \\ 3 \\ 3 \\ 3 \\ 3 \\ 3 \\ 3 \\ 5 \\ 4 \end{bmatrix}$$

$$obj = sum\big(A(1{:}8,:)\big)$$

$$sense = '<<<<<<<<<==='$$

$$modelsense = 'min'$$

Como podemos ver, en este caso, las matrices "A" y "rhs" contienen las restricciones y "obj" contiene la función objetivo que está relacionada con las capacidades de los enlaces, que son concretamente las 8 primeras filas de la matriz "A".

Ahora solo quedaría añadir estos valores al modelo y después realizar la llamada a la función "gurobi", lo cual podemos hacerlo como mostramos a continuación:

$$model.A = sparse(A)$$

$$model.rhs = rhs$$

$$model.obj = obj$$

$$model.sense = sense$$

$$model.modelsense = modelsense$$

$$result = gurobi(model)$$

2.3. Criterios de elección

En sección vamos a comparar los tiempos de resolución del problema inicial con MATLAB y Gurobi mediante los algoritmos explicados en este capítulo. Para ello utilizaremos las topologías triángulo, pentágono, NSFNet y RedIris. Crearemos un conjunto de demandas de tráfico estáticas para cada topología de red y calcularemos los tiempos de resolución en segundos con ambos programas, como podemos ver en la Tabla 1.

	Triángulo	**Pentágono**	**NSFNet**	**RedIris**
MATLAB	0.265648 seg	0.774780 seg	10.450234 seg	324.150171 seg
Gurobi	0.007972 seg	0.003896 seg	0.003357 seg	0.004457 seg

Tabla 1: Comparación de tiempos de resolución

Podemos observar que los tiempos de resolución con Gurobi son infinitamente inferiores que los de MATLAB. De hecho, vemos que cuando la topología es de una extensión mayor y las demandas de tráfico aumentan, los tiempos de resolución de MATLAB aumentan considerablemente, mientras que el tiempo de resolución de Gurobi sigue siendo prácticamente despreciable.

Por lo tanto, a partir de ahora utilizaremos Gurobi para la implementación de los siguientes algoritmos ya que es la herramienta de mejor rendimiento para este tipo de problemas.

3.- ESTABLECIMIENTO ESTÁTICO DE CAMINOS DE LUZ

Después de haber hecho una introducción a RWA y de haber explicado de forma sencilla cómo podemos utilizar Gurobi, vamos a pasar a describir el problema SLE y posteriormente veremos cómo estructurarlo para poder resolverlo con Gurobi, tal y como hemos hecho en la creación de nuestro algoritmo.

3.1. Introducción a SLE

El problema de enrutamiento y asignación de longitud de onda para tráfico estático es conocido también como problema Static Lightpath Establisment (SLE). En este problema, las peticiones de caminos de luz se conocen de antemano y las operaciones de enrutamiento y asignación de longitud de onda se llevan a cabo fuera de línea. El objetivo típico es reducir al mínimo el número de longitudes de onda necesarias para configurar un determinado conjunto de caminos de luz en una topología física dada. Como alternativa a este objetivo, el problema dual es maximizar el número de conexiones que se pueden establecer (minimizar el bloqueo) para un número dado de longitudes de onda y un conjunto de solicitudes de conexión. El doble problema SLE plantea la cuestión de la equidad, en que las soluciones a este problema tenderán a establecer conexiones más cortas que atraviesan un menor número de enlaces que conexiones largas que pasan por un mayor número de enlaces.

El problema de maximizar el número de conexiones establecidas para un número fijo de longitudes de onda y un conjunto dado de peticiones de conexión se puede formular como un modelo de programación lineal entera (ILP del inglés Integer Linear Program). Para este problema se definen:

- N_{sd} : Número de pares origen-destino.
- L : Número de enlaces.
- W : Número de longitudes de onda por enlace.
- $m = \{m_i\}$, $i = 1, 2, ..., N_{sd}$: Número de conexiones establecidas para el par origen-destino i.
- ρ : Carga ofrecida (número total de solicitudes de conexión para ser enrutadas).
- $q = \{q_i\}$, $i = 1, 2, ..., N_{sd}$: Fracción de carga que llega para el par origen-destino i.
- P : Conjunto de caminos en el que una conexión se puede enrutar.
- $A = (a_{ij})$: Matriz de tamaño P x N_{sd} donde $a_{ij} = 1$ si la ruta i es para el par origen-destino j, y $a_{ij} = 0$ en caso contrario.
- $B = (b_{ij})$: Matriz de tamaño P x L donde $b_{ij} = 1$ si el enlace j está en la ruta i, y $b_{ij} = 0$ en caso contrario.
- $C = (c_{ij})$: Matriz de ruta y asignación de longitud de onda de tamaño P x W, donde $c_{ij} = 1$ si la longitud de onda j se asigna a la ruta i, y $c_{ij} = 0$ en caso contrario.

El objetivo de este problema es maximizar el número de conexione establecidas, $C_0(\rho,q)$. La formulación IPL sería la siguiente:

Maximizar: $C_0(\rho, q) = \sum_{i=1}^{N_{sd}} m_i$ (1)

sujeto a

$m_i \geq 0, \qquad i = 1,2,\ldots,N_{sd}$ (2)

$c_{ij} \in \{0,1\} \qquad i = 1,2,\ldots,P;\ j = 1,2,\ldots,W$ (3)

$C^T B \leq 1_{W\times L}$ (4)

$m \leq 1_W C^T A$ (5)

$m_i \leq q_i \rho, \qquad i = 1,2,\ldots,N_{sd}$ (6)

La ecuación (1) obtiene el número total de conexiones que se establecen en la red. La ecuación (4) especifica que una longitud de onda se puede utilizar como máximo una vez en un enlace, donde 1_{WxL} es una matriz de tamaño W x L cuyos elementos son la unidad. Por su parte, las ecuaciones (5) y (6) se aseguran de que el número de conexiones establecidas es menor que el número de conexiones requeridas, donde 1_W es la matriz de tamaño 1 x W cuyos elementos son la unidad.

3.2. Adaptación de SLE para resolver con Gurobi

El problema que nos surge después de formular las ecuaciones es sobre qué necesitamos para poder revolver el problema utilizando Gurobi. Es entonces cuando vemos que la solución está en modificar las ecuaciones (4) y (5) de modo que queden de la forma A x = B.

Nos damos cuenta de que obstáculo principal es reside en las incógnitas, "c_{ij}" , que en esta formulación residen en forma de matriz, por lo que tendremos que modificarlas para que tomen forma de vector. Y a partir de ahí modificaremos el resto de elementos de las ecuaciones en función a este vector y el formato A x = B Por lo tanto quedarían las nuevas ecuaciones (4) y (5):

(4') $$B_2 \begin{bmatrix} c_{11} \\ c_{21} \\ \vdots \\ c_{i1} \\ c_{12} \\ c_{22} \\ \vdots \\ c_{i2} \end{bmatrix} \leq 1_{WL\times 1}$$

(5')
$$A_2 \begin{bmatrix} c_{11} \\ c_{21} \\ \vdots \\ c_{i1} \\ c_{12} \\ c_{22} \\ \vdots \\ c_{i2} \end{bmatrix} \geq m^T$$

Donde A_2 y B_2 son las matrices modificadas A y B de las ecuaciones (4) y (5) respectivamente; $1_{WL\ x\ 1}$ es una matriz de tamaño WL x 1 cuyos elementos son la unidad; y m^T es la matriz m traspuesta.

Las matrices A_2 y B_2 tienen que ser modificadas de tal forma que cumplan con las ecuaciones anteriores. Por lo tanto, la matriz A_2 tendrá un tamaño N_{sd} x PW y la matriz B_2 tendrá un tamaño WL x PW. Por tanto las ecuaciones finales con las que podremos utilizar Gurobi para resolver el ejercicio, quedan de la siguiente forma:

$$\begin{bmatrix} B_1 & & 0 \\ & B_2 & \\ & \ddots & \\ 0 & & B_k \end{bmatrix} \begin{bmatrix} c_{11} \\ c_{21} \\ \vdots \\ c_{i1} \\ c_{12} \\ c_{22} \\ \vdots \\ c_{i2} \\ \vdots \\ c_{iW} \end{bmatrix} \leq 1_{WL \times 1}$$

donde k = 1, 2, ..., W y $B_k = \begin{bmatrix} B_{11} B_{21} \dots B_{i1} \\ B_{12} B_{22} \dots B_{i2} \\ \vdots \quad \vdots \quad \dots \quad \vdots \\ B_{1j} B_{2j} \dots B_{ij} \end{bmatrix}$.

$$[A_1\ A_2 \dots\ A_k] \begin{bmatrix} c_{11} \\ c_{21} \\ \vdots \\ c_{i1} \\ c_{12} \\ c_{22} \\ \vdots \\ c_{i2} \\ \vdots \\ c_{iW} \end{bmatrix} \geq m^T$$

donde k = 1, 2, ..., W y $A_k = \begin{bmatrix} A_{11} A_{21} \dots A_{i1} \\ A_{12} A_{22} \dots A_{i2} \\ \vdots \quad \vdots \quad \dots \quad \vdots \\ A_{1j} A_{2j} \dots A_{ij} \end{bmatrix}$.

Para ver cómo se verían afectadas estas ecuaciones en un caso concreto, utilizaremos el ejemplo de la Ilustración 16. Suponiendo W = 2 longitudes de onda, una conexión para cada demanda (m_i = 1, con i = 1, 2), las ecuaciones antes de ser modificadas quedarían de la siguiente forma:

$$(4) \qquad \begin{bmatrix} c_{11} c_{21} c_{31} c_{41} \\ c_{12} c_{22} c_{32} c_{42} \end{bmatrix} \begin{bmatrix} 1\,1\,0\,0\,0\,1\,0\,0 \\ 0\,0\,1\,0\,0\,0\,1\,1 \\ 1\,1\,0\,0\,0\,0\,0\,0 \\ 0\,1\,1\,1\,0\,0\,0\,0 \end{bmatrix} \leq \begin{bmatrix} 1\,1\,1\,1\,1\,1\,1\,1 \\ 1\,1\,1\,1\,1\,1\,1\,1 \end{bmatrix}$$

$$(5) \qquad [1\,1] \leq [1\,1] \begin{bmatrix} c_{11} c_{21} c_{31} c_{41} \\ c_{12} c_{22} c_{32} c_{42} \end{bmatrix} \begin{bmatrix} 1\,0 \\ 1\,0 \\ 0\,1 \\ 0\,1 \end{bmatrix}$$

Por lo tanto, modificando primero la matriz de incógnitas y posteriormente reestructurando las matrices, nuestras restricciones para el ejemplo quedarían de la siguiente manera:

$$(4') \qquad \begin{bmatrix} 1\,0\,1\,0\,0\,0\,0\,0 \\ 1\,0\,1\,1\,0\,0\,0\,0 \\ 0\,1\,0\,1\,0\,0\,0\,0 \\ 0\,0\,0\,1\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0\,0\,0\,0 \\ 0\,1\,0\,0\,0\,0\,0\,0 \\ 0\,1\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,1\,0\,1\,0 \\ 0\,0\,0\,0\,1\,0\,1\,1 \\ 0\,0\,0\,0\,0\,1\,0\,1 \\ 0\,0\,0\,0\,0\,0\,0\,1 \\ 0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,1\,0\,0\,0 \\ 0\,0\,0\,0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,0\,1\,0\,0 \end{bmatrix} \begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \\ c_{12} \\ c_{22} \\ c_{32} \\ c_{42} \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$(5') \qquad \begin{bmatrix} 1\,1\,0\,0\,1\,1\,0\,0 \\ 0\,0\,1\,1\,0\,0\,1\,1 \end{bmatrix} \begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \\ c_{12} \\ c_{22} \\ c_{32} \\ c_{42} \end{bmatrix} \geq \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Y por último, los elementos del modelo necesarios para resolver el ejercicio con Gurobi, quedarían así:

$$A = \begin{bmatrix} 1&0&1&0&0&0&0&0 \\ 1&0&1&1&0&0&0&0 \\ 0&1&0&1&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&0&0&1&0&1&0 \\ 0&0&0&0&1&0&1&1 \\ 0&0&0&0&0&1&0&1 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&1&0&0 \\ 1&1&0&0&1&1&0&0 \\ 0&0&1&1&0&0&1&1 \end{bmatrix} \quad rhs = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$obj = sum(A(1{:}16, :))$$

$$sense = '<<<<<<<<<<<<<<<<>>>'$$

$$modelsense = 'min'$$

Nótese que en este caso la función objetivo comprenderá las 16 primeras filas de la matriz A.

Observamos que objetivo es maximizar el número total de conexiones establecidas. Para ello debemos maximizar el número de conexiones establecidas para cada par origen-destino i, "m_i". Y como hemos podido deducir de la ecuación (5), esto se consigue minimizando el valor de las incógnitas "c_{ij}". Por lo tanto, el elemento "modelsense" adquiere el valor 'min', en lugar de 'max' como podíamos llegar a pensar en un primer momento.

Para finalizar solo tendríamos que añadir los elementos al modelo y realizar la llamada a la función "gurobi" como hemos hecho en el Apartado 2.2.2.

3.3. Experimento SLE con Gurobi

Crearemos un experimento, utilizando la interfaz de Gurobi para MATLAB, que consistirá en, a partir de un número total de demandas D, generar aleatoriamente el origen y el destino para cada demanda e intentar asignarles un camino a todas y cada una de las demandas.

Este ensayo tiene dos objetivos:

- Observar como varía la probabilidad de "not allocate", que es la probabilidad de no asignar caminos a todas y cada una de las demandas del sistema, en función del número de longitudes de onda.
- Observar como varía el tiempo medio en resolver el ejercicio, medido en segundos, en función del número de longitudes de onda.

Para este experimento utilizaremos un único camino posible para cada demanda, el camino más corto (con menos saltos) desde el origen de la demanda hasta su destino.

Emplearemos para esta prueba la "topología de prueba" y la "topología Telefónica". Utilizaremos un número específico de demandas y de longitudes de onda para cada topología, debido a la diferencia de tamaño que hay entre ellas.

Pasamos a ilustrar primero el experimento en la topología de prueba:

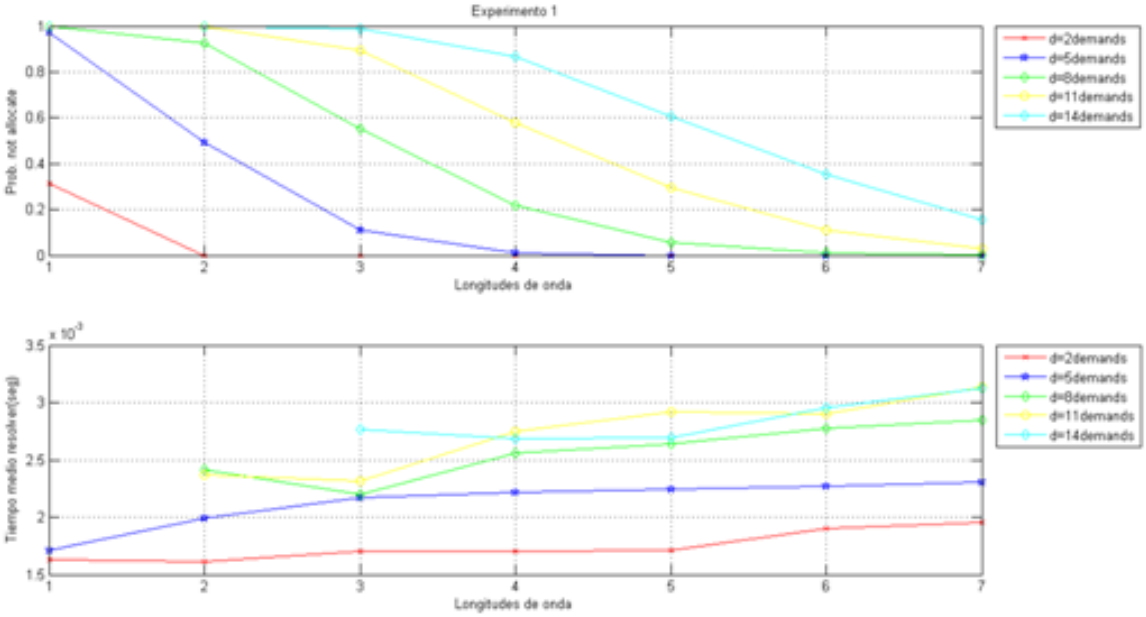


Ilustración 18: Experimento 1 topología de prueba

En la gráfica superior de la Ilustración 18 podemos ver la representación de la probabilidad de "not allocate" en función de las longitudes de onda. En la gráfica inferior advertimos la representación del tiempo medio en resolver el problema en función de las longitudes de onda.

Hemos utilizado los distintos números de demandas, N_d:

$$N_d = 2, 5, 8, 11 \text{ y } 14$$

y las distintas longitudes de onda, W:

$$W = 1, 2, 3, 4, 5, 6 \text{ y } 7$$

Lanzamos el ejercicios 10000 veces para cada una de las demandas con cada longitud de onda y dibujamos los resultados medios de la probabilidad de "not allocate" y del tiempo medio en resolver el problema.

En el gráfico superior podemos observar que la probabilidad de "not allocate" es mayor cuanto mayor es el número de demandas, es decir, cuantas más demandas hay que encaminar más difícil es poder encaminarlas todas juntas por la red. También nos damos cuenta de que esta probabilidad disminuye cuando aumentamos el número de longitudes de onda, esto se debe a que podemos asignar más demandas a un mismo enlace de la red, por ejemplo, con W = 2 por cada enlace se permite que pasen dos flujos de demandas (cada uno en una longitud de onda distinta), con W = 4 por cada enlace podemos transportar cuatro demandas, y así sucesivamente.

Podemos visualizar en la gráfica inferior que, por lo general, el tiempo en resolver el problema es mayor cuando más demandas hay en el sistema, esto se debe a que el programa debe hacer más iteraciones para resolver el ejercicio y las matrices son de dimensiones mayores. También es posible visualizar que tarda más tiempo en resolver el problema si utilizamos mayor número de longitudes de onda.

Podemos ver en el gráfico inferior que para 8 demandas (línea verde), 11 demandas (línea amarilla) y 14 demandas (línea azul claro) el trazo empieza en 2, 3 y 2 longitudes de onda respectivamente. Esto sucede porque la probabilidad de "not allocate" con longitudes de ondas menores es igual a 1, es decir, que es imposible asignar caminos a ese número de demandas con esas longitudes de onda en la topología de prueba.

Por otro lado, la topología Telefónica quedaría de la siguiente forma:

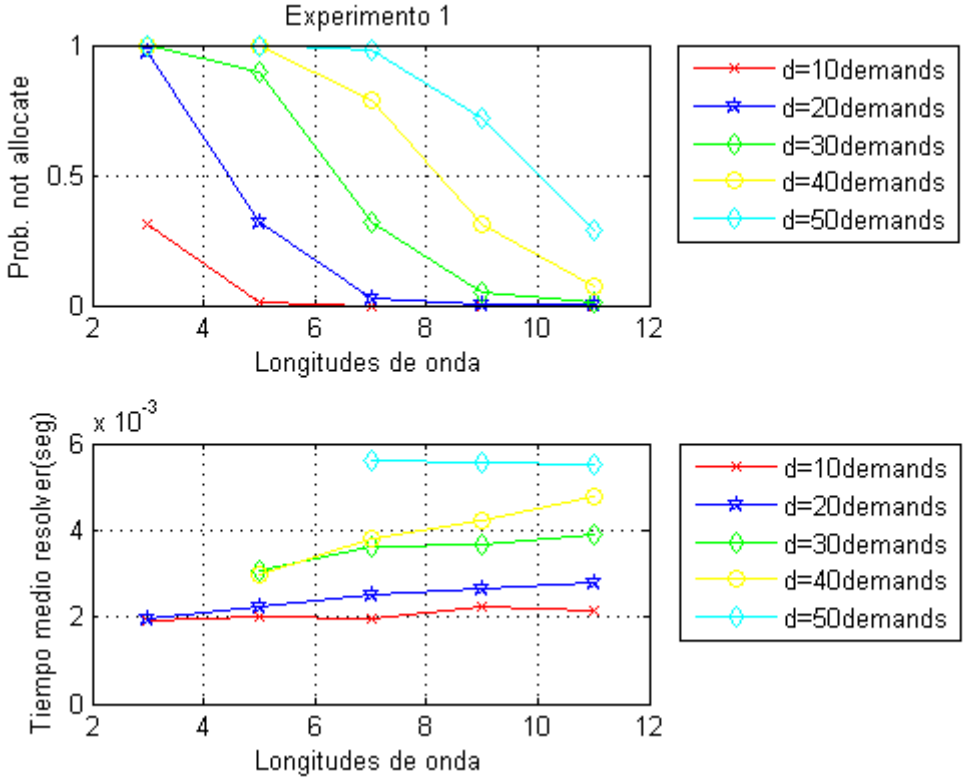


Ilustración 19: Experimento 1 Topología Telefónica

Para esta topología hemos utilizados los siguientes números de demandas, N_d:

$$N_d = 10, 20, 30, 40 \text{ y } 50$$

y las siguientes longitudes de onda, W:

$$W = 3, 7, 9, 11 \text{ y } 13$$

Este ejercicio es lanzado también 10000 veces para cada una de las demandas con cada longitud de onda y dibujamos los valores medios de la probabilidad de "not allocate" y del tiempo medio en resolver el problema.

La diferencia que queríamos encontrar con respecto al ejemplo de la topología de prueba es el tiempo medio en resolver el ejercicio. Este tiempo, en general, es mucho mayor en el ejemplo de la Ilustración 19, debido a que utilizamos mayor número de demandas y de longitudes de onda.

Pero podemos observar que tienen un intervalo común ambas figuras (línea verde de 8 demandas y línea amarilla de 11 demandas de la Ilustración 18 con la línea roja de 10 demandas de la Ilustración 19), donde coinciden las dos figuras aproximadamente en el número de demandas y longitudes de onda. En este intervalo observamos que, aunque a priori parezca contradictorio por las extensiones de las topologías, el tiempo de la topología Telefónica es considerablemente menor que en la topología de prueba.

Esto se debe a que realizar el ejercicio tantas veces (diez mil), creando demandas aleatorias, al tener un mayor número de nodos y de enlaces, es más fácil llevar a cabo el enrutamiento de todo el conjunto de demandas de cada realización.

3.4. Conclusiones

En este apartado hemos explicado la implementación de nuestro algoritmo que nos ha permitido solucionar el problema SLE y que además nos ha dejado corroborar varias conclusiones que esperábamos que se reflejasen en nuestros resultados.

Por un lado, hemos visto como aumenta la probabilidad de "not allocate" cuando aumentamos el número de demandas de tráfico. Esto era algo lógico y esperábamos comprobarlo con nuestro experimento, como así ha sido.

También hemos concluido que cuando incrementamos el número de longitudes de onda disponibles, podemos encaminar un mayor número de demandas de tráfico y por tanto, también desciende la probabilidad de "not allocate".

Otro dato que era de esperar es que el tiempo en resolver el problema es mayor cuando mayor es el número de demandas que hay en el sistema, esto se debe a que el programa debe hacer más iteraciones para resolver el ejercicio y las matrices son de dimensiones mayores.

Por último, nos hemos dado cuenta de que el tiempo en resolver el problema disminuye cuando utilizamos una topología de mayor extensión, debido a que es más sencillo enrutar todas las demandas por el sistema al tener un mayor número de enlaces.

4.- DIMENSIONAMIENTO DE RED. COLOREADO DE GRAFOS

En este capítulo se contará cómo hemos afrontado el siguiente objetivo: para una matriz de tráfico dada, buscar el menor número de longitudes de onda necesarias para satisfacer todas las demandas, dado un routing estático. Para ello hemos implementado un algoritmo utilizando el Graph Theory Toolbox de MATLAB que se basa en la Teoría de grafos y que resuelve este problema.

4.1. Introducción a coloreado de grafos

En Teoría de grafos, Graph Coloring o coloración de grafos es un apartado de etiquetado de grafos, que consiste en la asignación de etiquetas, de nominadas colores, a los elementos del grafo.

Este término se originó intentando asignar colores a los distintos países de un mapa, de manera que cada que las naciones que compartieran una frontera común no podían recibir el mismo color.

La coloración de grafos se divide en tres tipos:

- Vértice coloración: consiste en la asignación de colores a los vértices de un grafos de tal forma que dos vértices que compartan la misma arista tengan colores distintos.
- Arista coloración: consiste en la asignación de colores a las aristas de un grafo de tal manera que las aristas incidentes tengan colores distintos.
- Polinomio cromático: consiste en contar el número de formas distintas en que se puede colear un grafo utilizando número máximo de colores.

En este proyecto nos vamos a centrar en el tipo vértice coloración para resolver un experimento que explicaremos en la sección 4.3, para el cual utilizaremos un paquete de MATLAB llamado grTheory.

4.2. GrTheory y grColVer

GrTheory es un paquete que pertenece al Graph Theory Toolbox de MATLAB y está formado por una gran variedad de funciones que nos permiten resolver los problemas relacionados con la Teoría de grafos. Para el experimento, que comentaremos en el próximo apartado, vamos a utilizar la función perteneciente a este paquete denominada "grColVer".

La función "grColVer" nos permite resolver el problema de coloración de vértices del grafo. Es una función bastante simple de utilizar puesto que únicamente cuenta con un parámetro de entrada y un parámetro de salida.

La llamada a la función sería de la siguiente manera:

nCol = grColVer(E)

El parámetro de entrada "E" es una matriz con tantas filas como número de enlaces tenga la red y con únicamente dos columnas, donde colocaremos los dos nodos o vértices que componen cada enlace. De una forma un poco más clara, la matriz "E" lo que en realidad contiene es la representación del grafo.

Por otra parte, el parámetro de salida "nCol" será un vector de tamaño igual al número de nodos de la red. Cada posición del vector contendrá el número de colores de los vértices, es decir, cada número representa un color, siendo así el valor máximo del vector "nCol" el número mínimo de colores que necesitaríamos para resolver el problema.

A continuación veremos cómo adaptamos esta función para conseguir el mínimo número de longitudes de onda necesarias para transportar por la red un número de demandas dadas.

Como ya hemos mencionado anteriormente, la función del ejemplo de coloración de vértices es asignar un color a cada vértice de manera que, con el mínimo número de colores, no haya dos vértices que tengan mismo color y compartan una arista. Esto nos hace ver la similitud que hay con dos demandas que pasan por un mismo enlace ya que, al igual que con los vértices, no queremos que tengan la misma longitud de onda, si no que vayan por longitudes de onda distintas. Es entonces cuando se nos ocurre crear un grafo a partir de las demandas generadas aleatoriamente, donde los vértices serán nuestras demandas y las aristas unirán dos demandas que comparten al menos un enlace. De esta manera, el número mínimo de colores necesarios para que ningún vértice repita color, será nuestro mínimo número de longitudes de onda necesarias para encaminar todas las demandas a través de la red.

Ilustraremos la adaptación de este ejercicio con el siguiente ejemplo:

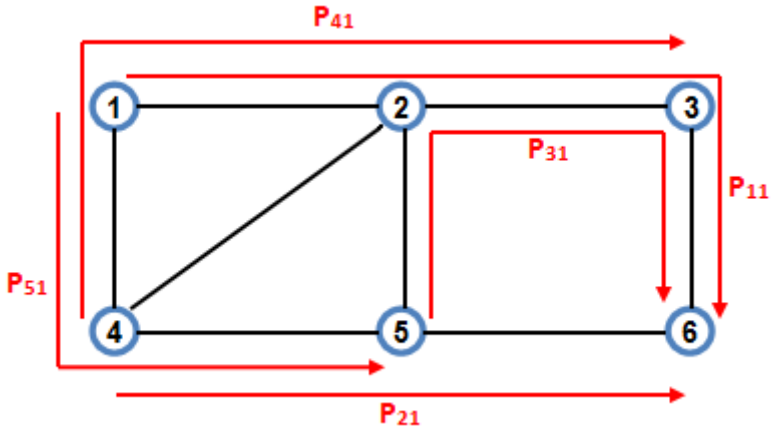


Ilustración 20: Topología de prueba con demandas

Tenemos cinco demandas con sus correspondientes caminos. Crearemos ahora el nuevo grafo auxiliar, que quedaría de la forma:

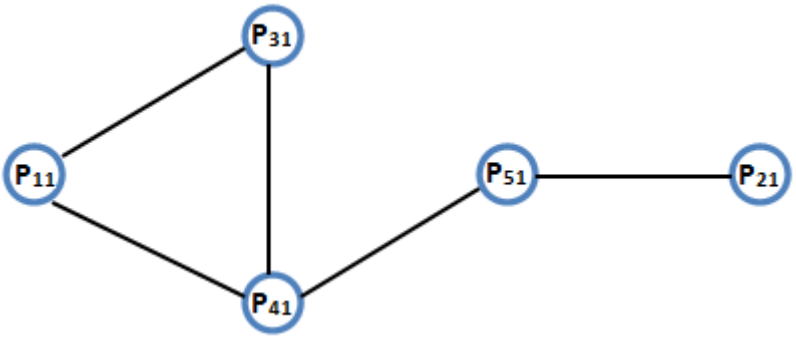


Ilustración 21: Grafo auxiliar

En este nuevo grafo unimos las demandas que comparten al menos un enlace de la topología de prueba. Por lo tanto, solo nos queda crear la matriz "E" y realizar la llamada a la función "grColVer":

$$E = \begin{bmatrix} 1 & 3 \\ 1 & 4 \\ 2 & 5 \\ 3 & 4 \\ 4 & 5 \end{bmatrix}$$

$$nCol = grColVer(E)$$

Como se puede deducir, los números 1, 2, 3, 4 y 5 de la matriz "E" corresponden a P_{11}, P_{21}, P_{31}, P_{41} y P_{51} re2spectivamente y simbolizan los enlaces del nuevo grafo. Por otra parte, el vector "nCol" contiene en cada posición el número de longitud de onda donde se transportará esa demanda, correspondiendo el número de la primera posición de "nCol" con el número de longitud de onda donde será transportada la primera demanda, el número de la segunda posición con la longitud de onda de la segunda

demanda, y así sucesivamente. Siendo el valor máximo de este vector, el número mínimo de longitudes de ondas de onda necesarias para transmitir todas las demandas al mismo tiempo.

4.3. Experimento grColVer

Crearemos un experimento, utilizando la función "grColVer" del Graph Theory Toolbox de MATLAB, para adaptar el ejemplo de coloración de vértices y conseguir el número mínimo de longitudes de onda que serán necesarias para poder encaminar por la red un número de demandas creadas aleatoriamente.

Para este experimento utilizaremos un único camino posible para cada demanda, el camino más corto (con menos saltos) desde el origen de la demanda hasta su destino.

Emplearemos para este ensayo la "topología de prueba". Utilizaremos un número específico de demandas para esta topología.

Pasamos a ilustrar el experimento en la topología de prueba:

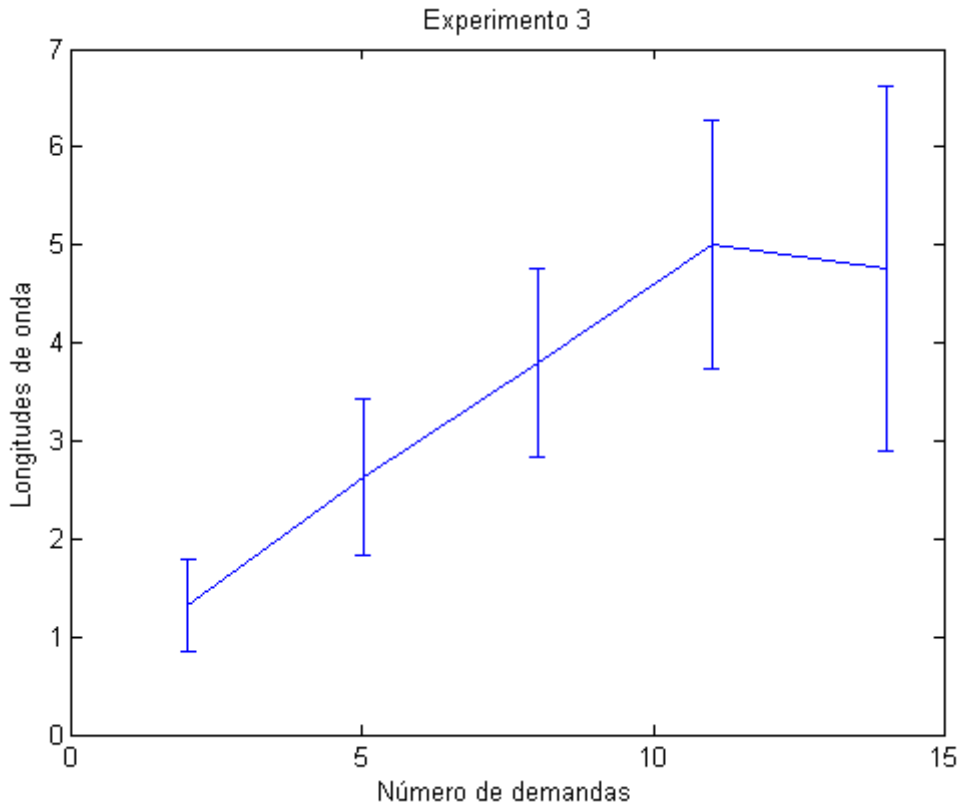


Ilustración 22: Experimento 3 Topología de prueba

En la Ilustración 22 podemos ver la representación de la variación del número de longitudes de onda en función del número de demandas. Se presenta la media ± 1 desviación típica. Por ejemplo, para 5 demandas, son necesarias 2.6 longitudes de onda (con desviación típica ± 0.7).

Hemos utilizado los distintos números de demandas, N_d:

$$N_d = 2, 5, 8, 11 \text{ y } 14$$

Lanzamos el ejercicio 100 veces para cada número de demandas y dibujamos los resultados medios y la desviación típica del número mínimo de longitudes de onda necesarias para satisfacer todas las demandas.

Podemos observar que como era de esperar, cuanto mayor es el número de demandas, más longitudes de onda son necesarias para satisfacer las demandas. Esto se debe a que cuantas más demandas hay, más fácil es que pasen demandas por un mismo enlace, y por tanto, necesitemos más longitudes de onda.

También podemos darnos cuenta de que a medida que aumenta el número de demandas aumenta la desviación típica de longitudes de onda. Esto sucede dependiendo de las demandas, al utilizar muchas demandas y lanzar el ejercicio tantas veces, habrá veces que coincidirán las demandas de tal manera que no se comparta un enlace muchas veces y se necesiten pocas longitudes de onda y otras veces, en cambio, muchas demandas compartirán un mismo enlace y serán necesarias una gran cantidad de longitudes de onda.

4.4. Conclusiones

En este apartado de dimensionamiento de red, nuestro algoritmo nos ha permitido solucionar el problema de asignación de longitudes de onda y también sacar diversas conclusiones respecto a nuestro problema.

Por un lado, como ya pudimos observar en el apartado anterior, se corrobora que cuanto mayor es el número de demandas de tráfico en el sistema, mayor es el número mínimo de longitudes de onda necesarias para satisfacer todas esas demandas. Que aumente el número de demandas significa que habrá un mayor número de demandas que compartirán un mismo enlace y que, por tanto, sean necesarias más longitudes de onda.

Por otro lado, en este capítulo, observamos que cuando se incrementan el número de demandas de tráfico, también aumenta la desviación típica de longitudes de onda. Esto se debe a la generación de demandas de tráfico aleatorias, que en ocasiones será necesario un menor número de longitudes de onda y en otras ocasiones, necesitaremos un número mayor de longitudes de onda.

5.- Heurística para la asignación de recursos a demandas de tráfico

En este capítulo vamos a tratar la implementación de una simulación de tráfico no estático que hemos realizado en MATLAB, donde las demandas llegan a un sistema, son procesadas y posteriormente salen del sistema. Para ello haremos una breve introducción a conceptos básicos relacionados y pasaremos a exponer unos experimentos donde veremos los resultados de esta implementación.

5.1. Introducción a la heurística

El término heurística proviene de la palabra griega "heuriskein", que significa encontrar o descubrir. Se utiliza en el campo de la optimización para caracterizar un determinado tipo de métodos de resolución de problemas. Hay un gran número de problemas difíciles que surgen en la práctica y que es necesario resolver de manera eficiente, lo que ha promovido el desarrollo de procedimientos eficaces, en un intento de encontrar buenas soluciones, incluso si no son óptimas. Estos métodos, en los que la velocidad del proceso es tan importante como la calidad de la solución obtenida, se llaman algoritmos heurísticos o aproximados.

A diferencia de los métodos exactos, que garantizan dar una solución óptima del problema, los métodos heurísticos sol tratan de dar una buena, pero no necesariamente óptima solución. Sin embargo, el tiempo empleado por un método exacto para encontrar una solución óptima a un problema complejo, se encuentra en un mayor orden de magnitud que el método heurístico. Por lo tanto, a menudo se recurre a métodos heurísticos para resolver problemas de optimización reales.

Además de la necesidad de encontrar buenas soluciones a los problemas difíciles en un tiempo razonable, hay otras razones para utilizar métodos heurísticos, entre los que queremos destacar:

- No se conoce un método para resolver el problema óptimamente.
- Aunque hay un método exacto para resolver el problema, no se puede implementar en el hardware disponible.
- El método heurístico es más flexible que el método exacto, lo que permite, por ejemplo, la incorporación de condiciones que son difíciles de modelar.
- El método heurístico se utiliza como parte de un procedimiento global que garantiza encontrar la solución óptima a un problema.

Un buen algoritmo heurístico debe cumplir con las siguientes propiedades:

- Una solución se puede conseguir con un esfuerzo computacional razonable.
- La solución deber ser casi óptima (con alta probabilidad).

- La probabilidad de obtener una mala solución debe ser baja.

5.2. Introducción a teoría de colas

La teoría de colas es el análisis matemático de los fenómenos de las líneas de espera o colas. En este modelo matemático los clientes llegan al sistema de manera aleatoria, requiriendo recursos de servicio. A su llegada, se les hace esperar en la cola hasta que sea su turno para ser atendidos. Una vez servidos, se supone que abandonan el sistema.

Para estos modelos estaremos interesados en determinar medidas de rendimiento, como por ejemplo, el número medio de clientes en el sistema (o en la cola) y el tiempo medio que un cliente gasta en el sistema (o pasa esperando en la cola).

Se trata de una herramienta de gran importancia en las áreas de análisis y diseño de sistemas de telecomunicaciones, redes de ordenadores o call centers (centros de atención telefónica.

Este modelo parte de unos parámetros principales:

- K : número de servidores o capacidad del sistema (por ejemplo: 4 servidores)
- λ : número medio de clientes que llegan al sistema por unidad de tiempo (por ejemplo: 10 clientes/minuto).
- μ : tasa de servicio de clientes de cada servidor (por ejemplo: 6 clientes/minuto).

A partir de los cuales podemos definir:

- Tiempo medio entre llegadas: 1/λ
- Tiempo medio de servicio: 1/ μ
- Carga media por unidad de tiempo: λ/μ
- Factor de utilización: $\rho = \frac{\lambda/\mu}{K}$

En estos modelos la capacidad limitada de los recursos del sistema causa congestión, retrasos y/o pérdida de clientes. Si un sistema tiene la capacidad suficiente de procesar la carga de trabajo que llega, se dice que el sistema es estable. Por el contrario, si no tiene la capacidad suficiente de procesado, la congestión aumenta indefinidamente y se denomina como un sistema inestable.

5.3. Experimento heurístico de demandas

En este apartado vamos a ver a través de un experimento cómo varía la probabilidad de demandas no satisfechas (demandas no satisfechas divido de total de demandas) en función de la carga media por unidad de tiempo, para un número variable de

longitudes de onda y un número fijo de caminos más cortos posibles para cada demanda. Veremos los resultados para las topologías de prueba y Telefónica.

El experimento consiste en generar demandas con origen y destinos aleatorios e ir sirviendo estas demandas según lleguen al sistema por su camino/s más corto/s en función del número de longitudes de ondas disponibles.

Para ambas topologías llegarán demandas con una distribución exponencial de media $1/\lambda$, donde λ tomará valores desde 1 hasta 160, añadiendo cada vez una unidad, es decir, 159 valores de λ. Esto significa que llegarán desde 1 a 160 clientes al sistema por unidad de tiempo.

Estas demandas serán servidas según una distribución exponencial de media $1/\mu$, donde μ será igual a 8, es decir, el sistema servirá 8 clientes por unidad de tiempo.

Tomamos estos valores de λ y μ para que así la carga media por unidad de tiempo (λ/μ) tome valores entre 0.125 y 20.

A continuación podemos ver el caso de la topología de prueba y topología Telefónica, donde utilizaremos un total de 10000 demandas para cada valor de λ. Es decir, llegarán al sistema 10000 demandas con una tasa de llegada de 1 cliente por unidad de tiempo, otras 10000 demandas con una tasa de llegada de 2 clientes por unidad de tiempo, y así sucesivamente hasta llegar a una tasa de llegadas de 160 clientes por unidad de tiempo. Las 10000 demandas tendrán el mismo tiempo de servicio para todos los distintos valores de λ.

Empezaremos con el supuesto de la topología de prueba, para la que tomaremos las longitudes de onda, W:

$$W = 1, 3 \text{ y } 4$$

Así para el caso donde cada demanda únicamente tiene un posible camino desde tu nodo origen hasta su nodo destino (el camino más corto), veríamos la siguiente representación:

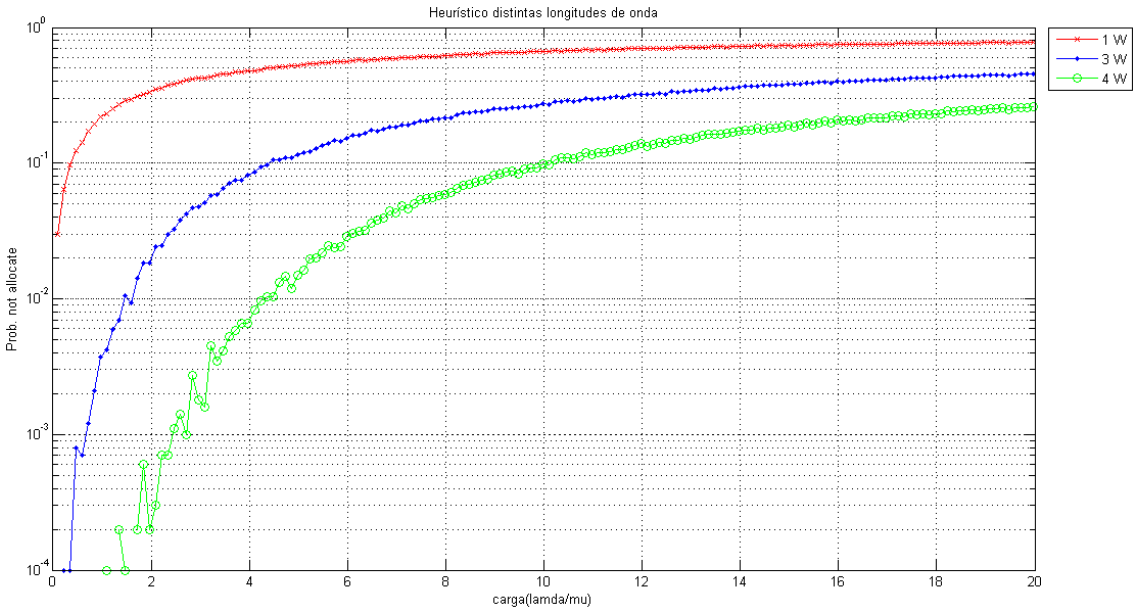


Ilustración 23: Experimento 4 Topología de prueba con 1 KSP

Podemos observar en la Ilustración 23 que, independientemente del valor de W, la probabilidad de demandas no satisfechas aumenta según va aumentando el valor de la carga media por unidad de tiempo. Esto ocurre porque fijamos el valor de μ y vamos incrementando el valor de λ, es decir, cada vez tenemos mayor número de llegada de demandas al sistema y la misma capacidad de satisfacer demandas. Por lo tanto, era de esperar que al llegar más demandas y tener la misma tasa de servicio esta probabilidad, o porcentaje de demandas no satisfechas, fuera cada vez mayor.

También podemos darnos cuenta, como era de esperar, que para un número mayor de longitudes de onda, la probabilidad de demandas no satisfechas es menor. Vemos que para una longitud de onda (línea roja) el porcentaje de demandas no satisfechas es mayor, para todos los valores de la carga, que cuando utilizamos 3 longitudes de onda (línea azul). Y a su vez cuando utilizamos 4 longitudes de onda (línea verde) tenemos una probabilidad menor que en cuando utilizamos 1 y 3 longitudes de onda. Esto se debe a que en cada enlace del sistema podemos utilizar mayor número de longitudes de onda y se pueden satisfacer mayor número de demandas, lo que provoca que la probabilidad de demandas no satisfechas disminuya.

Representamos el mismo experimento pero esta vez con los 3 posibles caminos más cortos para cada demanda:

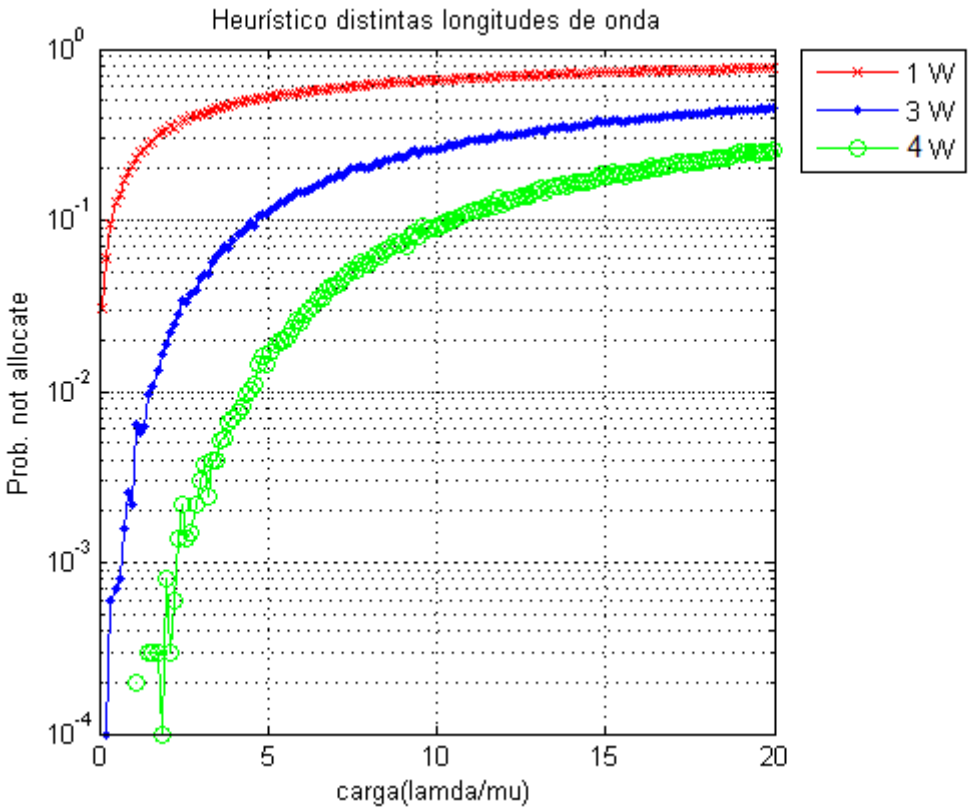


Ilustración 24: Experimento 4 Topología de prueba con 3 KSP

Y también para el caso donde tenemos los 4 caminos más cortos posibles para cada demanda:

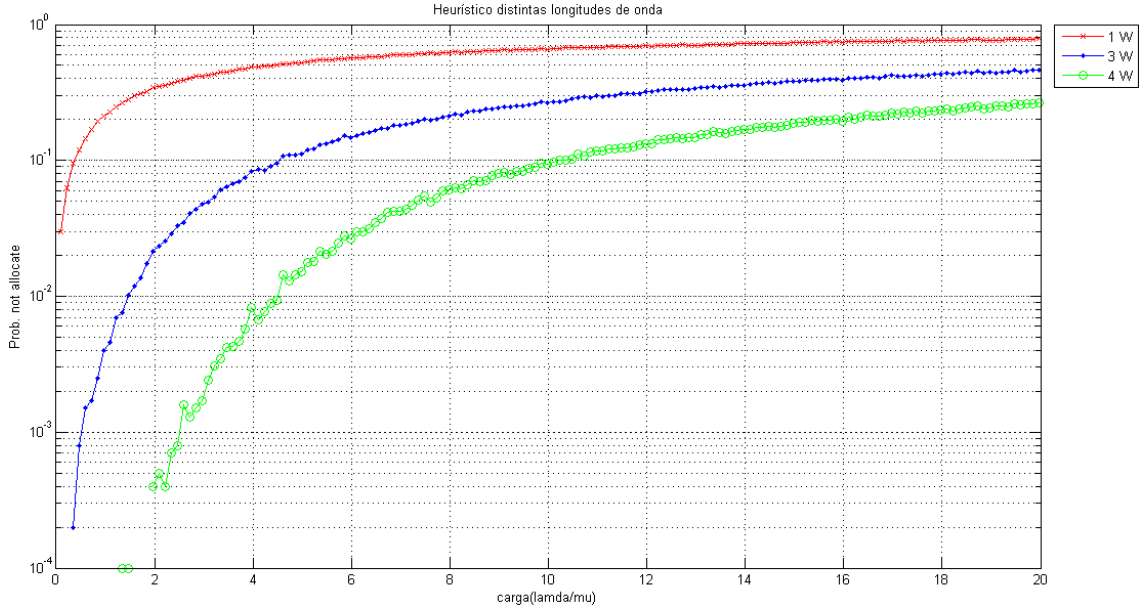


Ilustración 25: Experimento 4 Topología de prueba con 4 KSP

Podemos observar que las gráficas son prácticamente idénticas, donde las probabilidades de demandas no satisfechas en cada caso son exactamente iguales. Esto se debe a que al tener tantas demandas en una topología tan pequeña, aunque tengamos más caminos posibles para una demanda, los enlaces de esos caminos alternativos ya están ocupados por otras demandas y no pueden ser utilizados. Esto supone, que a efectos prácticos, estemos utilizando siempre un único camino para cada demanda.

Para el supuesto de la topología Telefónica, tomaremos las longitudes de onda, W:

$$W = 1, 3 \text{ y } 5$$

Así para el caso donde cada demanda únicamente tiene un posible camino desde tu nodo origen hasta su nodo destino (el camino más corto), veríamos la siguiente representación:

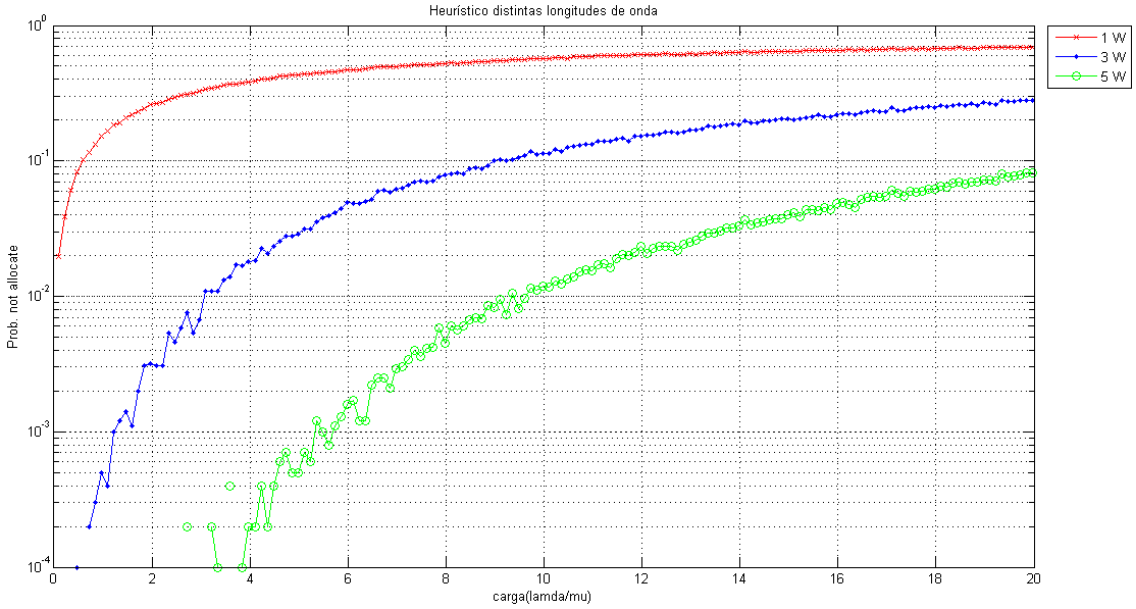


Ilustración 26: Experimento 4 Topología Telefónica con 1 KSP

Como podemos distinguir en la Ilustración 26, la probabilidad de demandas no satisfechas, con respecto a la 23 (Topología de prueba con un camino más corto posible), desciende ligeramente. Esto se debe a que la topología Telefónica es una red más grande y tiene muchos más enlaces por los que encaminar las demandas, lo que provoca que podamos servir un mayor número de demandas y disminuya el porcentaje de demandas no satisfechas.

Esto podemos comprobarlo fijándonos en las líneas roja (W=1) y azul (W=3) de las Ilustraciones 23 y 26. Estas líneas representan las mismas longitudes de onda en ambas gráficas y vemos que ambas líneas toman menores valores de la probabilidad de demandas no satisfechas independientemente del valor que adquiera la carga.

Como con la topología de prueba, para la topología Telefónica probamos el experimento con varios caminos más cortos posibles. En el caso de en que podemos utilizar los 3 caminos más cortos posibles para cada demanda, tenemos la siguiente representación gráfica:

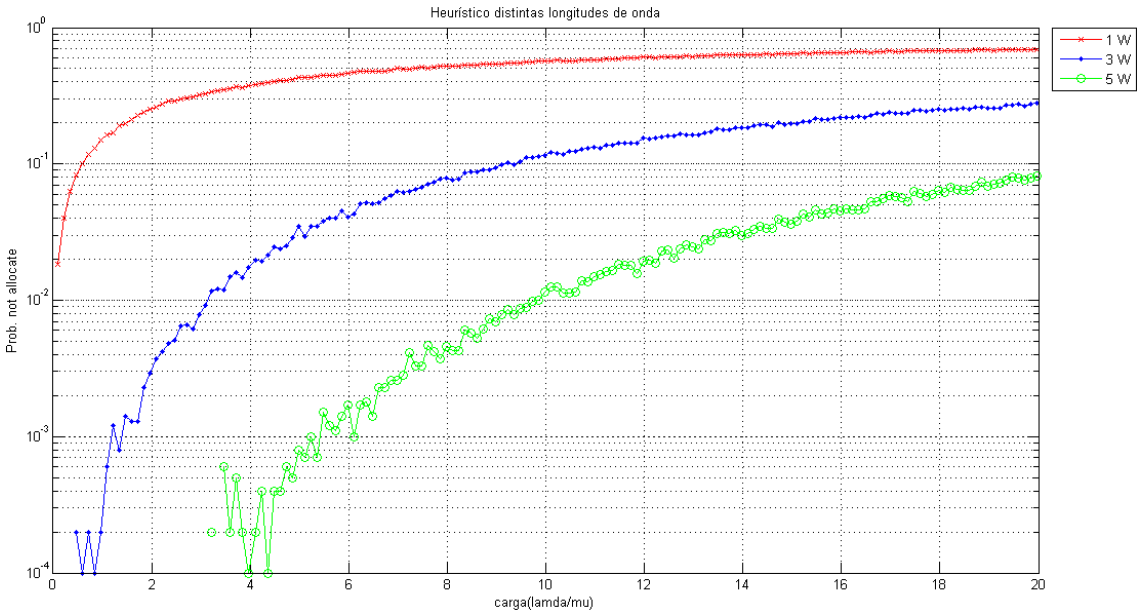


Ilustración 27: Experimento 4 Topología Telefónica con 3 KSP

Y también para el caso donde tenemos los 5 caminos más cortos posibles para cada demanda:

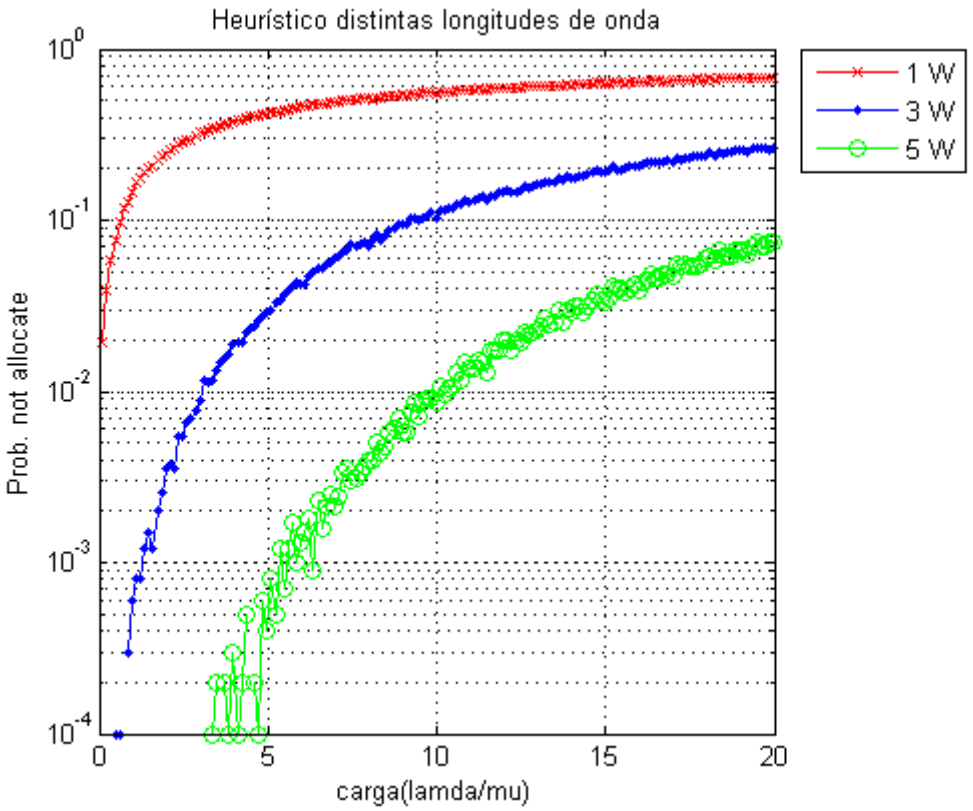


Ilustración 28: Experimento 4 Topología Telefónica con 5 KSP

Podemos observar, al igual que para la topología de prueba, que las gráficas son prácticamente idénticas, donde las probabilidades de "not allocate" en cada caso son exactamente iguales. Esto se debe a que al tener tantas demandas en una topología tan pequeña, aunque tengamos más caminos posibles para una demanda, los enlaces de esos caminos alternativos ya están ocupados por otras demandas y no pueden ser utilizados. Esto supone, que a efectos prácticos, estemos utilizando siempre un único camino para cada demanda.

5.4. Experimento heurístico de distintos k caminos más cortos

En este apartado, al igual que en el anterior, vamos a ver a través de un experimento cómo varía la probabilidad de demandas no satisfechas en función de la carga media por unidad de tiempo, pero esta vez mantenemos fijo el número de longitudes de onda y variamos el número de caminos más cortos posibles para cada demanda. Veremos los resultados para las topologías de prueba y Telefónica.

El experimento consiste en generar demandas con origen y destinos aleatorios e ir sirviendo estas demandas según lleguen al sistema por su camino/s más corto/s en función del número de longitudes de ondas fijado.

Los valores de λ, μ y carga serán los mismos que para el experimento del apartado anterior. Utilizaremos también 10000 demandas con nodos origen y destino generados aleatoriamente y las utilizaremos para cada valor de λ.

Empezaremos con el supuesto de la topología de prueba, para el que utilizaremos el siguiente número de caminos más cortos o KSP (K Shortest Path), para cada demanda:

KSP = 1, 3 y 4

Así para el caso donde únicamente dispusiésemos de una longitud de onda, tendríamos la siguiente representación gráfica:

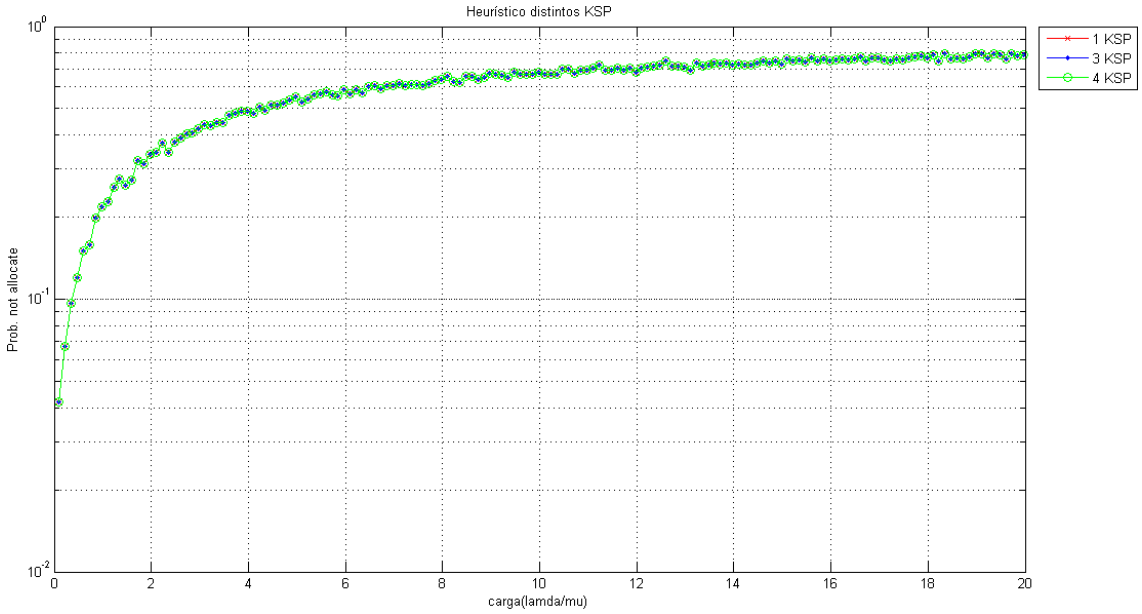


Ilustración 29: Experimento 5 Topología de prueba con 1 W

En la Ilustración 29 podemos observar que la probabilidad de demandas no satisfechas aumenta según va aumentando el valor de la carga media por unidad de tiempo. Esto ocurre porque fijamos el valor de μ y vamos incrementando el valor de λ, es decir, cada vez tenemos mayor número de llegada de demandas al sistema y la misma capacidad de satisfacer demandas. Por lo tanto, era de esperar que al llegar más demandas y tener la misma tasa de servicio, el porcentaje de demandas no satisfechas fuera cada vez mayor.

También podemos ver que las 3 líneas (línea roja, azul y verde) se superponen. Esto nos reafirma nuestra observación del apartado anterior, donde veíamos que las gráficas para distintos KSP eran idénticas. Esto, como comentamos anteriormente, se debe a que llegan al sistema un número muy elevado de demandas, 10000, por lo que

si no somos capaces de asignar una ruta a cada demanda a través de su camino más corto, no vamos a poder asignarle otro camino puesto que los enlaces van a estar ya ocupados. Esto hace que encaminemos por la red el mismo número de demandas y por tanto la probabilidad de demandas no satisfechas sea la misma.

Representamos el mismo experimento pero esta vez con 3 longitudes de onda disponibles:

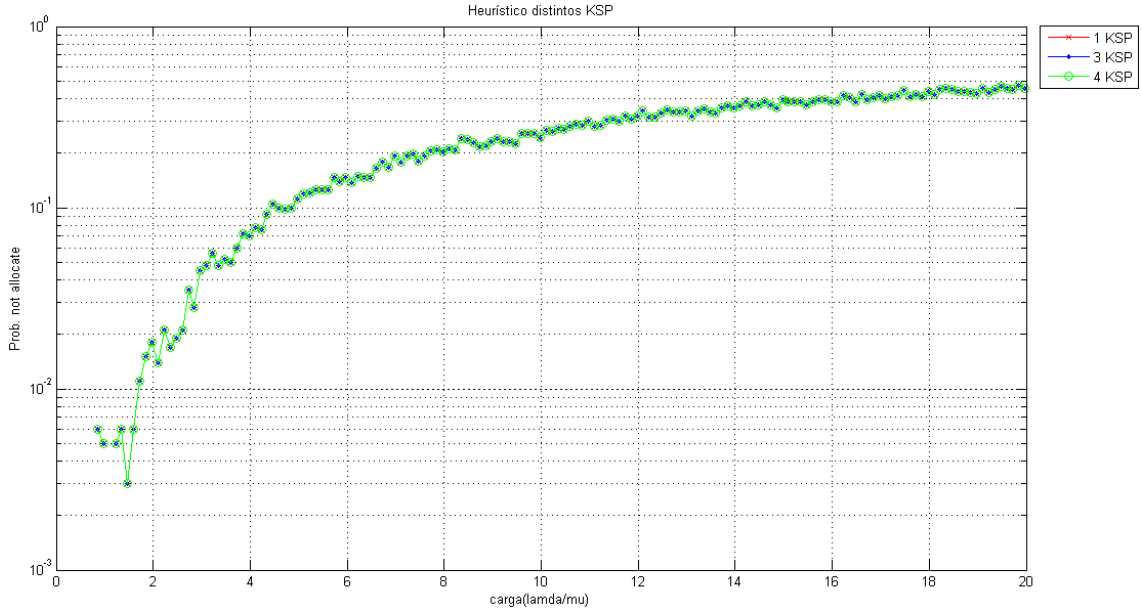


Ilustración 30: Experimento 5 Topología de prueba con 3 W

Y también para el caso donde tenemos 5 longitudes de onda disponibles:

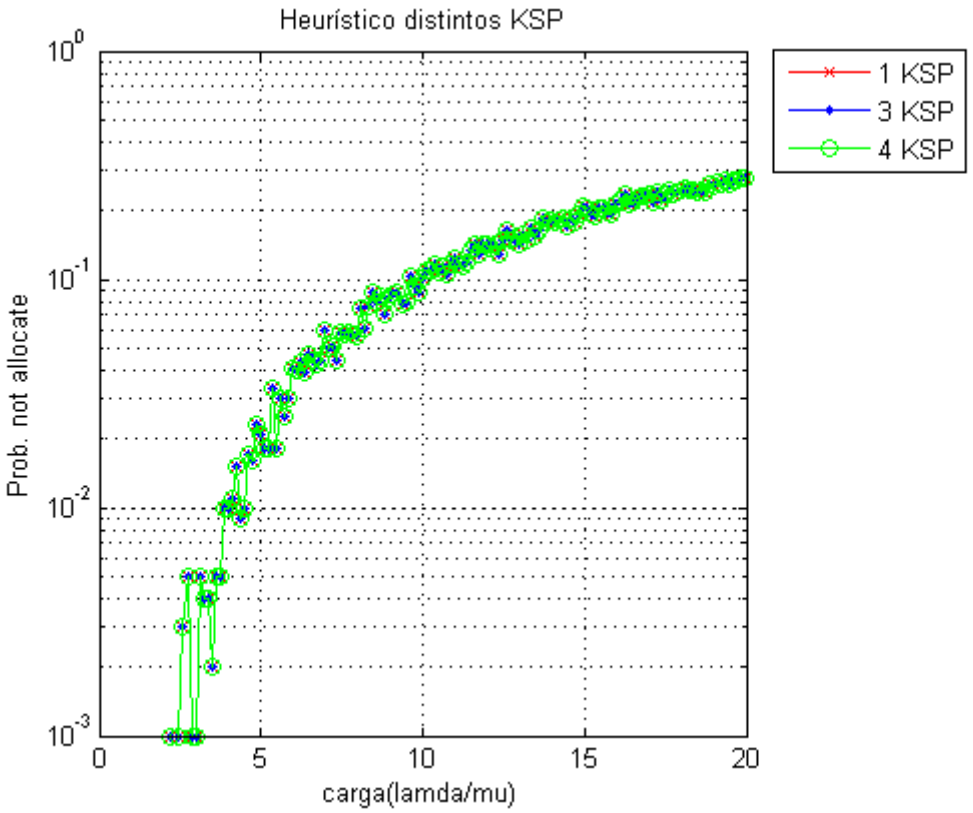


Ilustración 31: Experimento 5 Topología de prueba con 5 W

Como podemos observar y reafirmar los resultados del apartado 5.3, la probabilidad de demandas no satisfechas disminuye cuando aumentamos el número de longitudes de onda, independientemente del valor de la carga. Esto se debe a que en cada enlace del sistema podemos utilizar mayor número de longitudes de onda y se pueden satisfacer mayor número de demandas, lo que provoca que el porcentaje de demandas no satisfechas disminuya.

Para el supuesto de la topología Telefónica, tomaremos los siguientes números de caminos más cortos posibles para cada demanda, KSP:

KSP = 1, 3 y 4

Así para el caso donde únicamente dispusiésemos de una longitud de onda, tendríamos la siguiente representación gráfica:

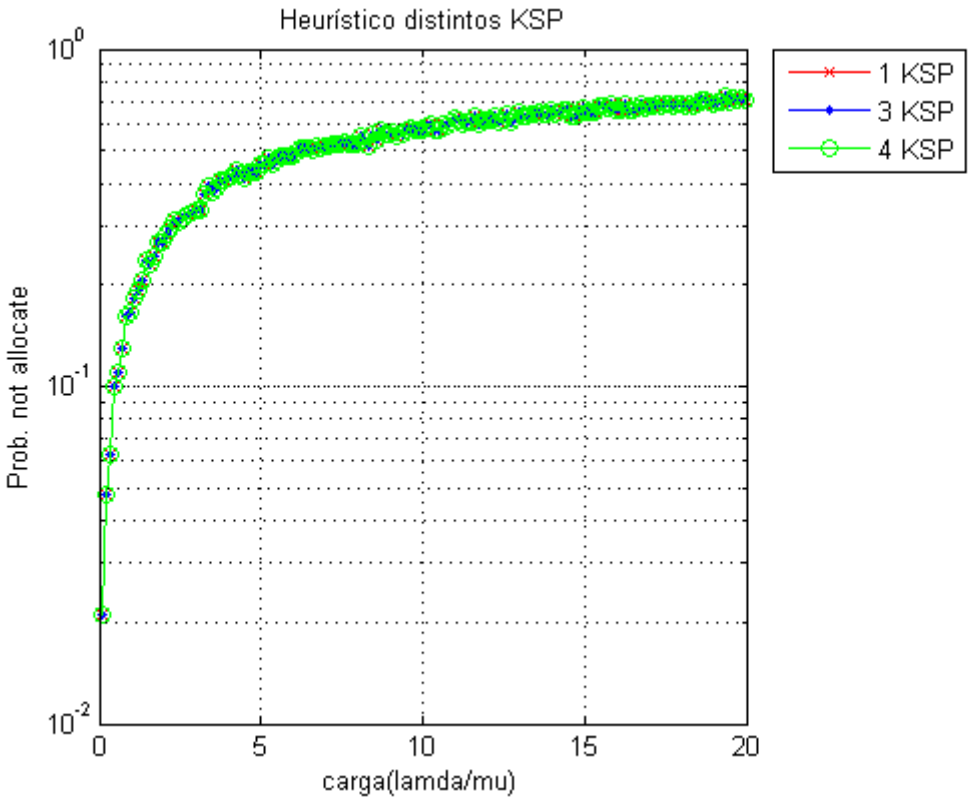


Ilustración 32: Experimento 5 Topología Telefónica 1 W

Observamos en la Ilustración 32 que para la topología Telefónica, las tres líneas también se superponen. Por lo tanto, el número de demandas sigue siendo muy elevado también para esta topología y efectos prácticos, independientemente del número de KSP que tengamos, si no fija el camino más corto cada demanda, no fijará otro camino con mayor número de enlaces.

También distinguimos al comprar las Ilustraciones 29 y 32, donde tenemos una única longitud de onda, mismo número de demandas y de KSP y lo único que cambia es la topología utilizada, que la probabilidad de demandas no satisfechas es ligeramente menor para la topología Telefónica (Ilustración 32). Esto confirma lo observado en el apartado 5.3, donde veíamos que la topología Telefónica es una red más grande y tiene muchos más enlaces por los que encaminar las demandas, lo que provoca que podamos servir un mayor número de demandas y disminuya el porcentaje de demandas no satisfechas.

Como con la topología de prueba, para la topología Telefónica probamos el experimento para distintos números de longitudes de onda. En el caso de en que podemos utilizar 3 longitudes de onda, tenemos la siguiente representación gráfica:

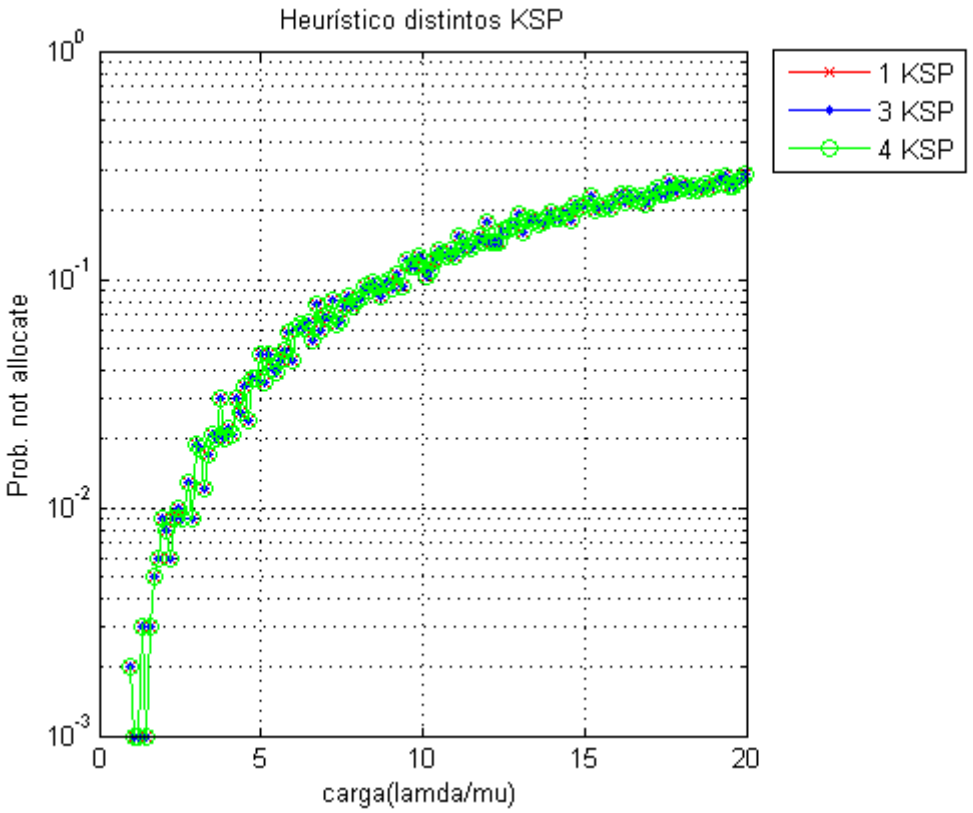


Ilustración 33: Experimento 5 Topología Telefónica 3 W

Y también para el caso donde tenemos 5 longitudes de onda disponible:

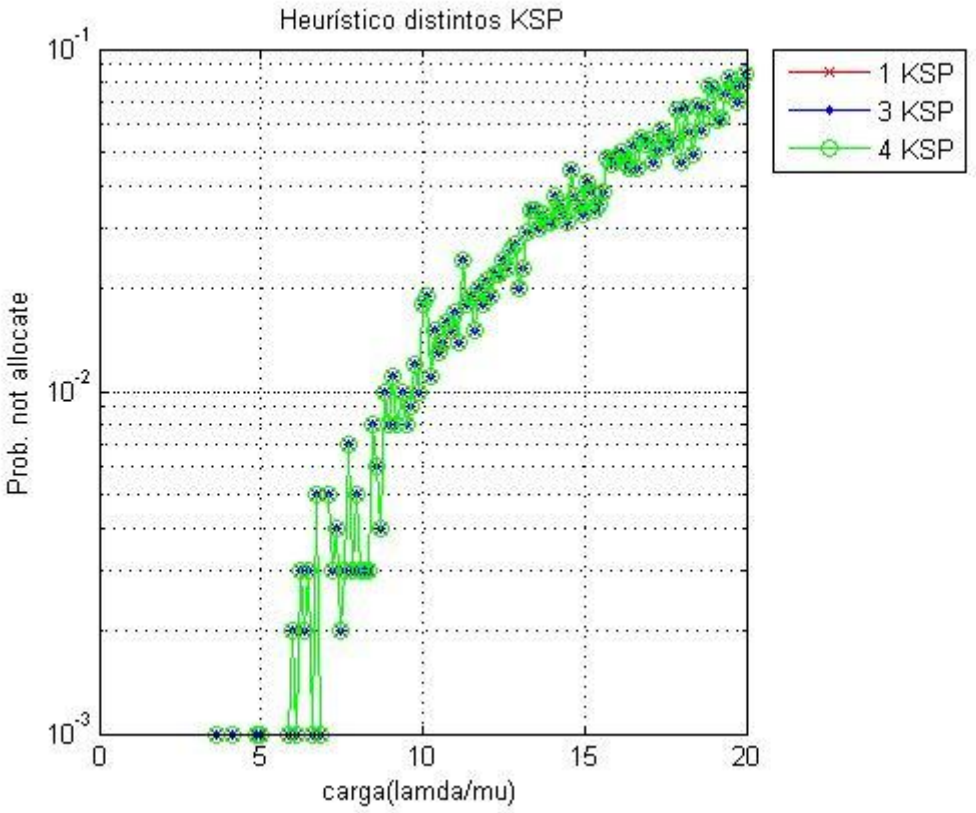


Ilustración 34: Experimento 5 Topología Telefónica 5 W

Podemos observar como esperábamos, al igual que para la topología de prueba, que la probabilidad de demandas no satisfechas disminuye cuando aumentamos el número de longitudes de onda, independientemente del valor de la carga. Esto se debe a que en cada enlace del sistema podemos utilizar mayor número de longitudes de onda y se pueden satisfacer mayor número de demandas, lo que provoca que el porcentaje de demandas no satisfechas disminuya.

5.5. Conclusiones

En este capítulo hemos puesto a prueba un método heurístico de teoría de colas, de donde podemos sacar diversas conclusiones.

Por un lado, hemos demostrado que, como era de esperar, si utilizamos un mayor número de longitudes de onda, podremos satisfacer un mayor número de demandas debido a que por cada enlace de la red podrán transmitirse tantas demandas como longitudes de ondas utilicemos. Por tanto, a mayor número de longitudes de onda, más demandas satisfechas y menor probabilidad de demandas no satisfechas.

Por otro lado, comprobamos que para un mismo supuesto, una topología con mayor número de enlaces es capaz de satisfacer un mayor número de demandas que una red con menor número de enlaces. Por tanto, a mayor número de enlaces, mayor número de demandas satisfechas y menor porcentaje de demandas no satisfechas.

Por último, hemos podido darnos cuenta de que cuando llega a un sistema un número muy elevado de demandas, no importa el número de caminos más cortos que puedan tener las demandas, puesto que generalmente si no fija su camino en la ruta más corta (con menor número de enlaces), no fijará otro camino. Esto se debe a que al tener muchas demandas, el resto de enlaces van a estar ocupados, y si una demanda no logra fijar su camino más corto, es más difícil que fije otro utilizando un mayor número de enlaces.

6.- CONCLUSIONES Y TRABAJOS FUTUROS

En este apartado vamos a comentar las conclusiones globales del Trabajo Fin de Grado y los posibles trabajos futuros que se podrían desarrollar como continuación de este proyecto.

6.1. Conclusiones del TFG

Tras la realización de varios algoritmos a lo largo del Trabajo Fin de Grado, hemos podido a un gran número de conclusiones interesantes.

Los dos primeros algoritmos han sido implementados para comparar el tiempo de resolución del problema inicial en MATLAB y en Gurobi, con distintas topologías de red. Estos algoritmos nos han permitido llegar a la conclusión de que Gurobi es una mejor herramienta que MATLAB, para la resolución de problemas basados en inecuaciones con matrices de grandes dimensiones, puesto que llega a la misma solución, la correcta, con un tiempo de resolución mucho menor.

El siguiente algoritmo implementado resuelve el problema SLE de enrutamiento estático.

Gracias a la realización de este algoritmo, hemos podido comprobar como aumenta la probabilidad de "not allocate" cuando aumentamos el número de demandas de tráfico. También hemos visualizado que cuando incrementamos el número de longitudes de onda disponibles en un sistema, podemos encaminar un mayor número de demandas de tráfico por ese sistema y, por tanto, también desciende la probabilidad de "not allocate".

Otro resultado que era de esperar, y que hemos podido corroborar con este algoritmo, es que el tiempo en resolver el problema es mayor cuando el número de demandas que hay en el sistema es mayor. Esto es debido a que el algoritmo debe realizar más iteraciones para resolver el ejercicio y las matrices tienen dimensiones mucho mayores.

Para finalizar las conclusiones obtenidas de este algoritmo de enrutamiento estático, nos damos cuenta de que el tiempo en resolver el problema disminuye cuando utilizamos una topología de mayor extensión, y hemos deducido que se debe a que es más sencillo enrutar todas las demandas de tráfico por el sistema al tener este un mayor número de enlaces por los que encaminar las demandas.

A continuación, nuestro trabajo fue generar un algoritmo que resolviera el problema de asignación de longitudes de onda. Este programa lo basamos en la Teoría de grafos y nos ha permitido corroborar, por un lado, que cuanto mayor es el número de demandas de tráfico en una red, mayor es el número mínimo de longitudes de onda que son necesarias para satisfacer todas esas demandas. Que aumente el número de demandas de tráfico significa que habrá un número más elevado de demandas que compartirán un mismo enlace y que, por tanto, se necesitarán más longitudes de onda.

Por otro lado, gracias a este algoritmo, observamos que al incrementar el número de demandas de tráfico en un sistema, aumenta la desviación típica de longitudes de onda necesarias para satisfacer todas las demandas. Esto se debe a que la generación de demandas de tráfico es aleatoria, lo que provoca en ocasiones que varias demandas compartan enlaces y que sea necesario un mayor número de longitudes de onda y en otras ocasiones, no se comparta el mismo enlace tantas veces y necesitemos un número menor de longitudes de onda.

Para finalizar el estudio, implementamos dos algoritmos heurísticos basados en la Teoría de colas, para resolver un problema de demandas de tráfico dinámico, de los cuales además hemos podido sacado diversas conclusiones. Con estos algoritmos, hemos podido corroborar como con experimentos anteriores, que si utilizamos un mayor número de longitudes de onda, podremos satisfacer un número mayor de demandas de tráfico, puesto que por cada enlace del sistema óptico podrán transmitirse tantas demandas de tráfico como longitudes de ondas empleemos. Por lo tanto, a mayor número de longitudes de onda, mayor número de demandas satisfechas y menor porcentaje de demandas no satisfechas.

Por otro lado, comprobamos que para un mismo supuesto de demandas de tráfico, un sistema que tiene mayor número de enlaces es capaz de satisfacer un mayor número de demandas de tráfico que una red con menor número de enlaces. Por lo tanto, podemos concluir que a mayor número de enlaces, mayor número de demandas de tráfico satisfechas y, como consecuencia, menor porcentaje de demandas no satisfechas.

Por último, hemos podido darnos cuenta, con estos dos algoritmos heurísticos, de que al llegar a un sistema un número muy elevado de demandas de tráfico, no importa el número de caminos más cortos a los que puedan optar estas demandas, puesto que, generalmente, si no fija su camino en la ruta más corta (con menor número de enlaces), no fijará otro camino. Esto es debido a que al tener muchas demandas de tráfico, el resto de enlaces van a estar ocupados, y si una demanda no logra fijar su camino más corto, es más difícil que fije otro camino utilizando un mayor número de enlaces.

6.2. Trabajos futuros

En este apartado se comentan varios trabajos que podrían llevarse a cabo como continuación de este Trabajo Fin de Grado. Este conjunto de trabajos pueden ser:

- Extender el análisis de los resultados y conclusiones de todos los algoritmos implementados utilizando otras topologías de red. Para ello se podrían utilizar topologías más extensas o topologías de otros países.
- Realizar un estudio utilizando matrices de tráfico real y comparar los resultados obtenidos. Para ello sería necesario conocer los datos reales de tráfico de alguna operadora de telecomunicaciones.
- Implementación en lenguaje de programación Java o C, de una herramienta con interfaz gráfica de usuario. Esta herramientas nos podría permitir, por ejemplo, seleccionar la topología de red o el algoritmo que deseemos utilizar, así como las demandas de tráfico, el número de caminos más cortos posibles o el número de longitudes de onda.
- Búsqueda e implementación de algún algoritmo alternativo para el coloreado de grafos (Graph Coloring), que mejore el rendimiento del algoritmo implementado en este proyecto.
- Adaptar el algoritmo SLE para que resuelva el problema en redes elásticas, donde el ancho de banda es variable.

7.- BIBLIOGRAFÍA

B. Mukherjee: “Opticar WDM Networks”, Springer, 2006.

M. Maier: “Optical Switching Network”, Cambridge University Press, 2008.

R. Ramaswami, K. N. Sivarajan, G. H. Sasaki: “Optical Network: A Practical Perspective”, Morgan Kaufmann, 2009.

H. Zang, J. P. Jue, B. Mukherjee: “A Review of Routing and Wavelength Assignment Approaches for Wavelength-Routed Optical WDM Networks”. Optical Network Magazine, pp. 47-60, January 2000.

J. A. Hernández, P. Serrano: “Probabilistic models for computer networks: Tools and solved problems”, www.lulu.com, 2013.

S. M. Ross: “Introduction to Probability Models”, Academic Pr Inc, 2014.

O. C. Ibe: “Markov Processes for Stochastic Modeling”, Academic Pr Inc, 2008.

B. R. Haverkort: “Performance of Computer Communication Systems: A Model-Based Approach”, John Wiley & Sons, 1998.

E. K. P. Chong, S. H. Zak: “An Introduction to Optimization”, John Wiley & Sons, 2013.

F. S. Hillier, G. J. Liebermann: “Operations Research: Einführung”, Olderbourg, 2002.

M. Pioro, D. Medhi: “Routing, Flow, And Capacity Design In Communication And Computer Networks”, Morgan Kaufmann, 2004.

R. P. Grimaldi, D. J. Rothman: “Discrete and Combinatorial Mathematics”, Addison Wesley Pub Co Inc, 2003.

J. F. Kurose, K. W. Ross: "Computer Networking: A Top-Down Approach", Pearson, 2012.

MATLAB 2013. Consultado en http://www.mathworks.es/

Gurobi Optimizer. Consultado en http://www.gurobi.com/

ANEXO I:

PRESUPUESTO DEL PROYECTO

En este apartado vamos a exponer el presupuesto total de este proyecto, el cual dividimos en tres tipos de costes, según la naturaleza de los mismos: costes de componentes hardware, costes de componentes software y costes de personal. A continuación se muestran los distintos tipos de costes desglosados y posteriormente se hace el cálculo del coste total de realización del proyecto.

- **Costes de componentes hardware:** Comprenden los gastos originados en la compra del material necesario para el desarrollo completo del proyecto. El coste total del ordenador utilizado asciende a 425,62 €, que tras aplicarle el 21% correspondiente al IVA, la cifra asciende a 515,00 €. Si estimamos que el ordenador tiene un tiempo de vida de 4 años, el coste de amortización de un año sería de 128,75 €.

Componentes Hardware	Coste
Equipo informático con CPU Intel Core i5-2410M@2,3 Ghz. Sistema operativo Windows 7	128,75 €

Tabla 2: Costes de componentes hardware

- **Costes de componentes software:** Engloban los gastos ocasionados por la compra del software necesario para la realización completa del proyecto. Ha sido necesaria la compra de las licencias de MATLAB 2013 y Gurobi Optimizer para la realización del proyecto y del Paquete Microsoft Office Hogar y Empresas para la realización de la memoria.

Componentes Software	Costes
Licencia MATLAB 2013	2.000,00 €
Licencia Gurobi Optimizer	4.000,00 €
Paquete Microsoft Office Hogar y Empresas 2013	269,00 €

Tabla 3: Costes de componentes software

- **Costes de personal:** Abarcan todos los gastos generados por la contratación del personal encargado del desarrollo del proyecto. En este caso el desarrollo ha sido llevado a cabo por dos personas, alumno y tutor, que tomaremos como Ingeniero Junior e Profesor titular Doctor respectivamente.

Se considera un coste estimado de Hombre-Mes para un Ingeniero Junior sin experiencia de 3,000.00 € y para un Profesor titular Doctor con 9 años de experiencia investigadora de 5,500 €. Suponiendo una jornada de 8 horas diarias, corresponderían a 160 horas trabajadas a lo largo de un mes. Por tanto el coste estimado es de 18.75 €/hora como salario de un Ingeniero Senior y un coste de 34.375 €/hora como salario de un Profesor titular Doctor.

La jornada de trabajo del Ingeniero Junior ha sido de 3 horas diarias desde los mese de septiembre a enero, ambos incluidos. Lo que supone 60 horas mensuales trabajadas y un total de 300 horas durante este periodo. El tiempo de trabajo del Profesor titular Doctor comprende 20 tutorías de una hora de duración y 20 horas de dedicación a documentación y lectura de los avances y memoria del TFG, lo que supone un total de 40 horas trabajadas.

En la tabla 4 se detallan los costes de personal en función de las horas trabajadas.

Persona	**Número de horas**	**Coste/hora**	**Coste total**
Ingeniero Junior	300	18.75 €/hora	5,625.00 €
Profesor titular Doctor	40	34.375 €/hora	1,375.00 €

Tabla 4: Costes de personal

En la Tabla 5 queda reflejado el coste total del proyecto, comprendido por la suma de los costes de componentes hardware y software y los costes de personal.

Tipo de coste	Coste
Coste hardware	128,75 €
Coste software	6.269,00 €
Coste personal	7,000.00 €
Coste total	13,397.75 €

Tabla 5: Coste total del proyecto

ANEXO II:

ABSTRACT

In this project we study optical communication networks since these networks play a substantial role in current telecommunications systems. Particularly, we focus on Wavelength Division Multiplexing (WDM) as a transmission method. We study the routing and wavelength assignment (RWA) problem of current optical networks through inequalities systems which allow us to solve them optimally.

An optical network is a telecommunications network which performs some of its functionality by optical communication means. An optical communication is any form of communication which uses light as the transmission medium.

Wavelength Division Multiplexing is a technology that combines multiple signals over a single optical fiber through optical carriers with different wavelengths. WDM generally uses the light from a laser or LED.

Free space optical communication systems and fiber optics are the most common modern methods used in a large variety of applications. Free space optical communication systems are generally used in "last mile" communications. Fiber optics is the most used communication channel for optical communications, due to their benefits: immunity to electronic interference and low cost.

The fiber optic networks emerge as a solution for the increasing demands of the users and operators of telecommunication companies. They demand greater transmission capacity and security. We must also add a lower price.

When this demand to increase the transmission capacity between two points appears there were not available the necessary technologies to satisfy this demand. The only option was to install more fiber between these points, which meant a large investment of time and money. Then Wavelength Division Multiplexing (WDM) facilitated the solution. WDM could send a large number of signals over the same fiber, each with a different wavelength.

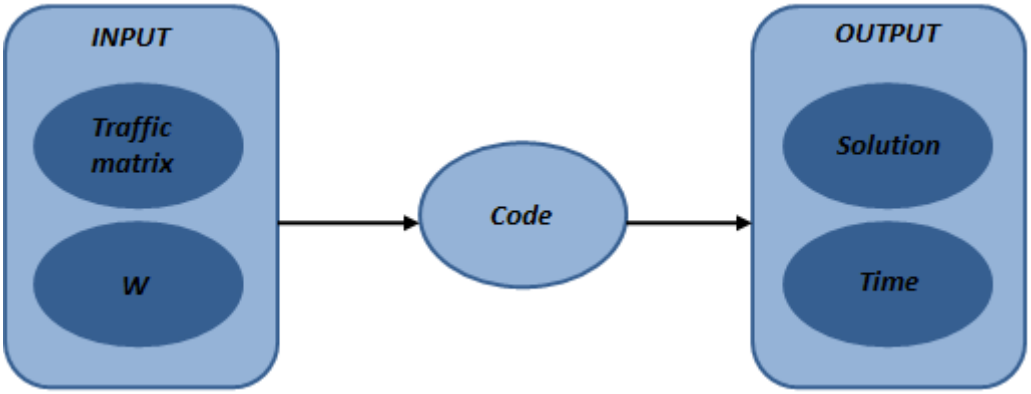
In a WDM optical network, end users can communicate with one another via optical channels, which are known as lightpaths. A lightpath is used to support a connection in a WDM optical network, and it may span multiple fiber links. A lightpath must occupy the same wavelength on all fiber links which it traverses.

In this project we will study the "Routing and Wavelength-Assignment" problem, or RWA problem. RWA is the problem of setting up lightpaths by routing and assigning a wavelength to each connection. Generally, connection requests can be of three types: static, dynamic or incremental.

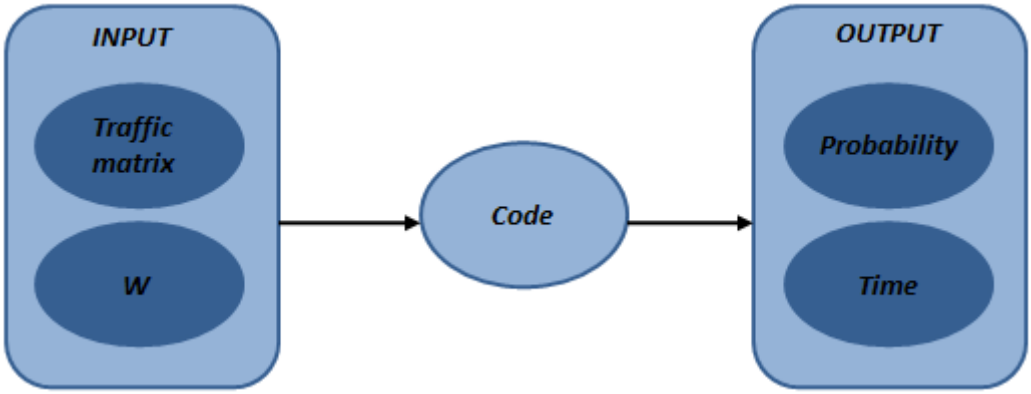
We use various network topologies to analyze all possible cases. These topologies have different sizes. We use the following network topologies: triangle topology, pentagon topology, test topology, RedIRIS topology, NSFNet topology and Telefónica topology.

We divide our study of optical systems in three phases: analysis phase, network dimensioning phase and design and network planning phase.

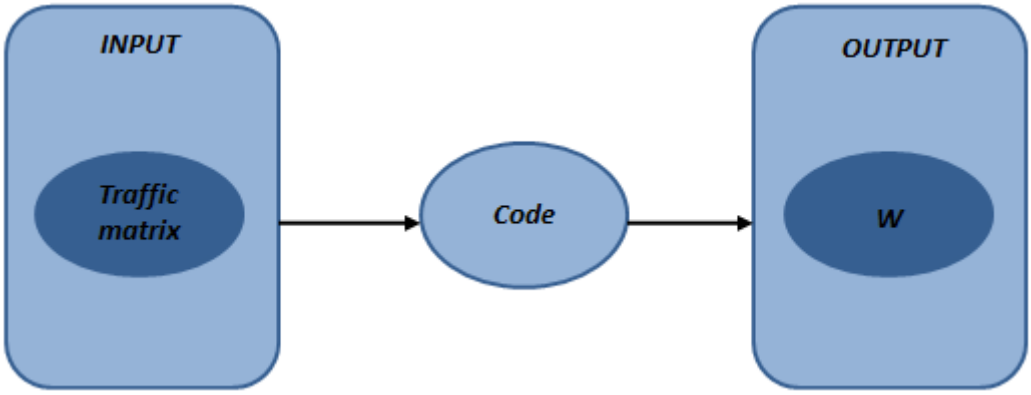
Chapter 2 contains the analysis part of the project. This section introduces the initial problem. We explain the algorithms created to solve this problem and compare the results. The implemented algorithms have the following structure:



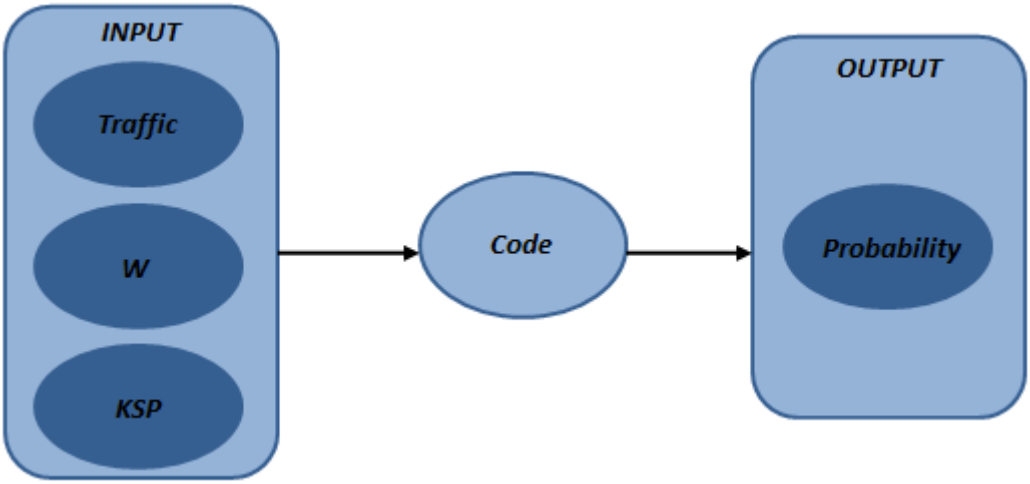
In section 3, we treat the Static Lightpath Establishment (SLE) problem. We explain the problem with notation and equations. Later we present the necessary changes in the equations in order to implement an algorithm that can solve the problem by using Gurobi and we explain the results. The algorithm has the following structure:



Chapter four contains the network dimensioning problem. This problem consists in assigning wavelengths and is based on the graphs theory. We explain the graphs theory and how we adapt it to our needs in order to implement an algorithm that solves our problem. This algorithm has the following structure:



In section 5, we explain the implementation of two heuristic algorithms based on the queueing theory. We introduce to the queueing theory and heuristic. Then we discuss the results of these two algorithms. Both programs take the following structure:



Following the completion of several algorithms in this project, we found many interesting conclusions.

The first two algorithms are implemented to compare the solution time of the initial problem with MATLAB and Gurobi. We performed a comparison with different network topologies. These algorithms have allowed us to conclude that Gurobi is a faster tool than MATLAB, to solve problems based on inequalities with large dimensions matrices. Gurobi gets the same solution but the time resolution is much lower than MATLAB.

We developed a third algorithm that solves the Static Lightpath Establishment problem.

Due to the implementation of this algorithm, we have seen that the "not allocate" probability increases when we increase the number of traffic demands. We have also displayed when we increase the number of available wavelengths in a system, we can route a larger number of traffic demands and, thus, also decreases the "not allocate" probability.

We have been able to corroborate another result with this algorithm. The result is that the problem resolution time is greater when the number of traffic demands in the system is higher.

To finish the conclusions of this SLE algorithm, we notice that the problem resolution time decreases when we use a topology with large dimensions. We deduced that this happens because it is easier to route all traffic demands on a system that has a greater number of links.

Then our work was to develop and implement an algorithm that would solve the wavelength assignment problem. This program is based on graph theory and has allowed us to corroborate that increasing the number of traffic demands in a network,

we need a higher minimum number of wavelengths to satisfy all these demands. Increasing the number of traffic demands mean there will be a higher number of demands that share the same link and, therefore, more wavelengths are needed.

In addition, we note that increasing the number of traffic demands offered to the system, increases the standard deviation of wavelengths required to satisfy all demands. We generate traffic demands randomly, which sometimes causes that various demands share the same link and we need a lot of wavelengths. In other occasions, the same link is not shared many times and we need fewer wavelengths.

To complete the study, we implemented two heuristic algorithms based on the queueing theory, to solve a problem of dynamic traffic demands. With these algorithms we have drawn different conclusions. We corroborated, as with previous experiments, that if we use a larger number of wavelengths, we can satisfy a larger number of traffic demands. Therefore, a greater number of wavelengths, as many satisfied demands and lower percentage of unsatisfied demands.

We have checked that for the same number of traffic demands, a system with more links can satisfy a larger number of traffic demands than other network with fewer links. Therefore, we can conclude that if we have more links we can satisfy more traffic demands and, as a result, we get a small percentage of unsatisfied demands.

Then, we have to realize with these two heuristic algorithms, that when a very high number of traffic demands arriving to a system, no matter the number of shortest paths to which these demands can choose, because if not fixed their way into the shortest path (with lower number of links), no fixed another way. This happens because if we have many traffic demands, the links are going to be busy, and if a traffic demand fails to fix its shortest path, it is more difficult to be fixed another path which has more links.

Finally, in budget section, we explain the necessary cost for the project. There are three types of costs: hardware components costs, software components costs and personnel costs. The total cost of this project is the sum of these costs and raises up to 13,397.75 €.

ANEXO III:

Fragmento de código de utilización de la función fmincon:

```
% 1.- CONSTRUIT LA MATRIZ A

A=zeros(NumEn,NumDem*NumKSP);

% Construir la matriz Adj
Adj=ones(N,N)*inf;

% Rellenar con 1s Adj

for i=1:size(T,1)
  Adj(T(i,1),T(i,2))=1; Adj(T(i,2),T(i,1))=1;
end

% Construir la matriz AdjCost
AdjCost=Adj;

% Rellenar A
dem=1;
for k=1:N
    for l=1:N
        if(k~=l)
            ksp=kShortestPath(AdjCost,k,l,NumKSP);
            for con=1:NumKSP
               tam=size(ksp{con},2) ;
               for con2=1:tam-1
                   d=ksp{con}(con2);
                   p=ksp{con}(con2+1);
                   for pos=1:NumEn
                      if(((T(pos,1)==d) && (T(pos,2)==p)) ||
((T(pos,1)==p) && (T(pos,2)==d)))
                           A(pos,dem)=1;
                      end
                   end
               end
               dem=dem+1;
            end
        end
    end
end

% 2.- CONSTRUIR LA MATRIZ DE COSTES (B)

C=T(:,3);

% 3.- CONSTRUIR LA MATRIZ Aeq

Aeq=zeros(NumDem,NumDem*NumKSP);
aux=1;

for j=1:size(Aeq,1)
    for cont=0:NumKSP-1
        Aeq(j,aux+cont)=1;
    end
    aux=aux+NumKSP;
end
```

```
% 4.- CONSTRUIR LA MATRIZ DE DEMANDAS (Beq)

H=TrafficMatrix_RedIris(:,3);

% 5.- CONSTRUIR LOS LÍMITES

Ub = [];
Lb = [];
x0 = zeros(1,NumDem*NumKSP);

% 6.-  OTRAS OPCIONES

nonlcon = [];
options = [];

% 7.- FMINCON
val=sum(A);
tic;
[x,fval,exitflag] =
fmincon(@funObj,x0,A,C,Aeq,H,Lb,Ub,nonlcon,options,A)
toc;
```

Fragmento de utilización de función gurobi:

```
% 1.- CONSTRUIT LA MATRIZ A

A=zeros(NumEn,NumDem*NumKSP);

% Construir la matriz Adj
Adj=ones(N,N)*inf;

% Rellenar con 1s Adj

for i=1:size(T,1)
  Adj(T(i,1),T(i,2))=1; Adj(T(i,2),T(i,1))=1;
end

% Construir la matriz AdjCost

AdjCost=Adj;

% Rellenar A
dem=1;
for k=1:N
    for l=1:N
        if(k~=l)
            ksp=kShortestPath(AdjCost,k,l,NumKSP);
            for con=1:NumKSP
               tam=size(ksp{con},2) ;
               for con2=1:tam-1
                   d=ksp{con}(con2);
                   p=ksp{con}(con2+1);
                   for pos=1:NumEn
                      if(((T(pos,1)==d) && (T(pos,2)==p)) ||
((T(pos,1)==p) && (T(pos,2)==d)))
                            A(pos,dem)=1;
                      end
                   end
```

```
                end
                dem=dem+1;
            end
        end
    end
end

% 2.- CONSTRUIR LA MATRIZ DE COSTES (B)

C=T(:,3);

% 3.- CONSTRUIR LA MATRIZ Aeq

Aeq=zeros(NumDem,NumDem*NumKSP);
aux=1;

for j=1:size(Aeq,1)
    for cont=0:NumKSP-1
        Aeq(j,aux+cont)=1;
    end
    aux=aux+NumKSP;
end

% 4.- CONSTRUIR LA MATRIZ DE DEMANDAS (Beq)

H=TrafficMatrix_NSFnet(:,3);

% 5.- GUROBI

for nd=1:NumDem
    signos(nd)='=';
end

for ne=1:NumEn
   signos(NumDem+ne)='<';
end

clear model;
model.A=sparse([Aeq; A]);
model.rhs=[H; C];
model.obj=sum(A);
model.sense= signos;
model.modelsense='min';
tic;
result=gurobi(model);
toc;
disp(result)
```

Fragnmento de algoritmo de graph coloring:

```
function [W,VZ] = simulateW(T, Dem, NumVeces, NumKSP)

    L=size(T,1); % Número de enlaces
    NumNod=max(max(T(:,1:2)));
    nW=zeros(1,NumVeces);

    for v=1:NumVeces
        E=zeros(Dem*L,2);
```

```
        G=[0 0];
        flg=0;
        for y=1:Dem
            a=randi(NumNod,1);
            b=randi(NumNod,1);
            while(b==a)
                b=randi(NumNod,1);
            end
            D(y,1)=a;
            D(y,2)=b;
        end

        N=size(D,1); % Número de demandas

        % Construir matriz de demandas
        B=zeros(N,L);
        Adj=ones(NumNod,NumNod)*inf;

        for k=1:L
            Adj(T(k,1),T(k,2))=1;
            Adj(T(k,2),T(k,1))=1;
        end

        ndem=1;
        con3=1;
        for l=1:N
            ksp=kShortestPath(Adj,D(l,1),D(l,2),NumKSP);
            for con=1:NumKSP
                tam=size(ksp{con},2);
                for con2=1:(tam-1)
                    s=ksp{con}(con2);
                    d=ksp{con}(con2+1);
                    for pos=1:L
                        if(((T(pos,1)==s) && (T(pos,2)==d)) ||
((T(pos,1)==d) && (T(pos,2)==s)))
                            B(ndem,pos)=1;
                            con3=con3+1;
                        end
                    end
                end
                ndem=ndem+1;
            end
        end

        % Contruir la matriz
        rep=0;
        for n=1:L
            for m=1:N-1
                if(B(m,n)==1)
                    for p=m+1:N
                        if(B(p,n)==1)
                            rep=rep+1;
                            E(rep,1)=m;
                            E(rep,2)=p;
                            %rep=rep+1;
                        end
                    end
                end
            end
        end
```

```
        % Poner repetidos a 0 en la matriz E
        for x=1:size(E,1)-1
            for y=x+1:size(E,1)
                if(E(x,1)==E(y,1) && E(x,2)==E(y,2))
                    E(y,1)=0;
                    E(y,2)=0;
                end
            end
        end

        % Construir matriz final
        ps=0;
        for z=1:size(E,1)
            if(E(z,1)~=0)
                ps=ps+1;
                G(ps,1)=E(z,1);
                G(ps,2)=E(z,2);
            end
        end

        % grColVer
        if(G(1,1)==0)
            flg=1;
        end

        if(flg~=1)
            nCol=grColVer(G);
            % Número de colores o longitudes de onda
            nW(v)=max(nCol);
        else
            nW(v)=1;
        end
    end
    W=mean(nW');
    VZ=std(nW');
end
```

Fragmento de algoritmos heurísticos:

```
T=Topologia_telefonica;
Ndemandas=10000;
NumKSP=[1,3,4];
W=5;
lambda=[0.8:1:160];
mu=8;
carga=lambda/mu;

prob = variosKSPnuevo(T,Ndemandas,NumKSP,W,lambda,mu);

codigos=['rx-';'b.-';'go-'];
leyendas={'1 KSP';'3 KSP'; '4 KSP'};

for i=1:size(NumKSP,2)

    semilogy(carga,prob(:,i),codigos(i,:)), hold on, grid on
    title('Heurístico distintos KSP')
    xlabel('carga(lamda/mu)')
    ylabel('Prob. not allocate')
```

```
    legend(leyendas,'Location','NorthEastOutside')

end


function prob = variosKSPnuevo(T,Ndemandas,NumKSP,W,lambda,mu)

L=size(T,1);
NumNod=max(max(T(:,1:2)));

for i=1:Ndemandas
    a=randi(NumNod,1);
    b=randi(NumNod,1);
    while(b==a)
        b=randi(NumNod,1);
    end
    D(i,1)=a;
    D(i,2)=b;
end

B=zeros(1,L);
Adj=ones(NumNod,NumNod)*inf;
for j=1:L
    Adj(T(j,1),T(j,2))=1;
    Adj(T(j,2),T(j,1))=1;
end

sl=size(lambda,2);
arrivals=zeros(sl,Ndemandas);
for ent=1:sl
    arrivals(ent,:)=cumsum(exprnd(1/lambda(ent),1,Ndemandas));
end
%arrivals
service_times = (exprnd(1/mu,1,Ndemandas));
departures=zeros(sl,Ndemandas);
for dep=1:sl
    departures(dep,:) = arrivals(dep,:)+service_times;
end
%departures
llegadas=sort(arrivals,2);
salidas=sort(departures,2);

tiempo_eventos=zeros(sl,2*Ndemandas);
heutipo_eventos=zeros(sl,2*Ndemandas);
colaSalida=zeros(sl,Ndemandas);
for nOrd=1:sl

[tiempo_eventos2,heutipo_eventos2]=ordenar(llegadas(nOrd,:),salidas(nO
rd,:));
    tiempo_eventos(nOrd,:)=tiempo_eventos2;
    heutipo_eventos(nOrd,:)=heutipo_eventos2;

colaSalida(nOrd,:)=ordenaSalidas(departures(nOrd,:),salidas(nOrd,:));
end

prob=zeros(sl,size(NumKSP,2));

for nv=1:size(NumKSP,2)
```

```
    for nl=1:sl
        caminos=zeros(Ndemandas,L);
        aux=0;
        aux2=1;
        aux3=0;
        aux4=0;
        aux5=0;
        rechazos=zeros(1,Ndemandas-1);
        rech=0;
        posKSP=0;

        for k=1:size(tiempo_eventos,2)

            % Eventos de llegada
            if(heutipo_eventos(nl,k)==1)
                aux=0;
                aux2=1;
                aux3=aux3+1;
                estado=0;
                Baux=zeros(W,L);
                posKSP=0;

                ksp=kShortestPath(Adj,D(aux3,1),D(aux3,2),NumKSP(nv));
                while(aux==0 && aux2<=NumKSP(nv))
                    posKSP=posKSP+1;
                    tam=size(ksp{posKSP},2);
                    for con=1:(tam-1)
                        s=ksp{posKSP}(con);
                        d=ksp{posKSP}(con+1);
                        for pos=1:L
                            if(((T(pos,1)==s) && (T(pos,2)==d)) ||
((T(pos,1)==d) && (T(pos,2)==s)))
                                Baux(k,pos)=1;

                                if(B(1,pos)==W)
                                    estado=1;
                                end
                            end
                        end
                    end
                    if(estado==0)
                        B=B+Baux(k,:);
                        caminos(aux3,1:L)=Baux(k,:);
                        aux=1;
                    end

                    aux2=aux2+1;
                    if(aux2==(NumKSP(nv)+1) && estado==1)
                        rech=rech+1;
                        rechazos(rech)=aux3;
                    end
                end

                % Eventos de salida
            elseif(heutipo_eventos(nl,k)==2)
                aux4=aux4+1;
                aux5=colaSalida(nl,aux4);
                flag=0;
                for m=1:size(rechazos,2)-1
                    if(aux5 == rechazos(m))
```

```
                        flag=1;
                    end
                end
                if(flag == 0)
                    for m=1:L
                        if(caminos(aux5,m)==1)
                            B(1,m)=B(1,m)-1;
                        end
                    end
                end
            end
        end
        prob(nl,nv)=sum(rechazos>0)/Ndemandas;
    end

end
end
```